University of Zurich[UZH]

Department of Informatics

Quadrotor Control for Accurate Agile Flight

Dissertation
submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by

Matthias Fässler
from Oberiberg, SZ

approved in April 2018 at the request of

Prof. Dr. Davide Scaramuzza, advisor
HDR. Dr. Antonio Franchi, examiner
Prof. Dr. Aníbal Ollero, examiner

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, April 4, 2018

Chairman of the Doctoral Board: Prof. Dr. Sven Seuken

What it meant to me will eventually
Be a memory of a time when
I tried so hard and got so far
But in the end, it doesn't even matter
— Linkin Park

To my friends and family.

Acknowledgements

I would like to thank my advisor Prof. Davide Scaramuzza for selecting me as a PhD student and giving me the opportunity and freedom to pursue many interesting ideas and projects.
During my time at the Robotics and Perception Group I learned a lot, gathered many valuable experiences, worked on interesting projects, and had many fun times. I am grateful to everyone who contributed to any of those things.
But also, I had to deal with many frustrating and disappointing things. Therefore, I would like to thank all my friends and family outside of RPG for their support and balancing the frustrations from work. These are especially my friends from the Feusisberg area, from the FC Feusisberg-Schindellegi, my parents and my brother.
I would like to thank the great friends I made within RPG, Flavio Fontana and Elias Mueggler, for the unique time we spent at the lab and in our spare time. It could have never been the same without you at the Lab!
This thesis is the result of many collaborations and fruitful discussions. I therefore want to thank Flavio Fontana, Elias Mueggler, Davide Falanga, Alessandro Simovic, Raphael Meyer, Michael Gassner, Suseong Kim, Titus Cieslewski, Julien Kohler, Philipp Foehn, Christian Forster, and Antonio Franchi for the active collaborations we had. Research would not be the same without fun distractions from work and I would therefore wish to express my gratitude to all the current and past members and visitors I did not directly work with: Guillermo Gallego, Jeffrey Delmerico, Henri Rebecq, Zichao Zhang, Antonio Loquercio, Elia Kaufmann, Mathias Gehrig, Alessio Zanchettin, Junjie Zhang, Matio Pizzoli, Manuel Werlberger, Reza Sabsevari, Andras Majdik, Gabriele Costante, Volker Grabe, Antonio Toma, Tamar Tolcachier, and the entire Grusig-Ässe-Tag group.
I also had the pleasure to work with great students, namely Benjamin Keiser, Karl Schwabe, Raphael Meyer, Adrian Rechy Romero, Maximilian Schulz, Astrid Schlestein, Michael Gassner, and Kevin Egger.
I would like to thank Prof. Aníbal Ollero and Dr. Antonio Franchi for reviewing my thesis and their valuable feedback.

Zurich, April 2018 M. F.

Abstract

Quadrotors are the most popular type of micro aerial vehicles. They gained their popularity due to their simple and robust mechanical design, which can be achieved from cheap off-the-shelf components. Furthermore, they are able to hover as well as takeoff and land vertically while still being able to perform agile maneuvers. Quadrotors are deployed in numerous applications, where sensors or payloads need to be moved to locations that are hard to reach or even unreachable for humans or ground robots. Among these applications are inspection of various infrastructure, monitoring and analysis in agriculture, surveillance, transportation, and aerial photography, which, together, form a multi-billion-dollar market.

Despite the many applications of quadrotors, their full potential, especially in terms of autonomy and agility, has not been exploited yet. Making quadrotors more autonomous and more agile brings the benefit of requiring fewer operators and completing tasks faster, which makes them more useful and increases their profitability. However, making quadrotor platforms more agile, e.g. by making them smaller, also makes them more difficult to control due to faster dynamics. Furthermore, during agile flight with high speeds and accelerations, aerodynamic effects, which are difficult to model and incorporate into the control, become relevant for the flight characteristics of quadrotors. Neglecting these effects does not affect their flight performance in near-hover flight but significantly reduces it during agile flights.

This thesis focuses on control methods that enable quadrotors to accurately track high-speed trajectories. The presented control methods comprise both methods for low-level attitude and body-rate control as well as high-level position control. They improve the trajectory tracking performance of a quadrotor by considering the dynamics of the single motors and considering linear rotor drag effects. Additionally, this thesis presents algorithms for operating quadrotors safely at their physical limits and recovering from failures. This thesis also presents contributions in the application of quadrotors within robot teams deployed in a search-and-rescue scenario as well as in enabling quadrotors to traverse narrow inclined gaps. The following is a list of contributions:

- Development of a quadrotor infrastructure for research on vision-based drone flight including hardware design and control algorithms.
- A method to recover a vision-based quadrotor from a lost state estimate, even in conditions with arbitrary attitude and high velocities, and re-establish stable

flight.

- The first proof that the dynamics of a quadrotor subject to linear drag effects are differentially flat. I propose a position controller that computes feed-forward control terms based on the differential flatness property, which enable accurate tracking of high-speed trajectories.
- Novel control method for quadrotor attitude and body-rate control that achieve more accurate reference tracking by considering the dynamics of the motors.
- Prioritizing input saturation method to achieve stable flight with quadrotors that are operated close to their physical limits where they reach motor saturations.
- The entire quadrotor control stack and required interfaces developed within this work is made available as open-source software.
- An algorithm for a quadrotor to efficiently collaborate with a ground robot in a search-and-rescue scenario.
- A pose estimator based on a monocular camera and infrared LEDs for relative localization of robot teams. This pose estimator was made available as open-source software.
- The first method that allows for a quadrotor to fly through narrow inclined gaps in an agile maneuver based only on onboard sensing and computation.

Zusammenfassung

Quadrokopter sind die am weitest verbreiteten Mikro-Fluggeräte. Sie erlangten ihre Popularität durch ihr einfaches aber robustes mechanisches Design, das mit günstigen Standardkomponenten erziehlt werden kann. Zudem sind sie fähig an Ort zu Schweben, senkrecht zu starten und landen aber auch sehr agile Manöver durchzuführen. Quadrokopter werden in zahlreichen Anwendungen, bei denen Sensoren oder Lasten an Orte die nur schwer oder gar unzugänglich für Menschen oder Bodenroboter sind, eingesetzt. Solche Anwendungen beinhalten unter Anderem Inspektionen diverser Infrastruktur, Überwachung and Analyse in der Landwirtschaft, allgemeine Luftüberwachung, Transport und Luftfotografie, die zusammen einen Multimillionen Markt bilden.

Trotz der vielen Anwendungsbereichen von Quadrokoptern ist deren volles Potential bei weitem noch nicht ausgeschöpft, vor allem bezüglich deren Automomie and Agilität. Selbstständigere und agilere Quadrokopter bringen die Vorteile das weniger Bediener benötigt werden und dass Missionen schneller ausgeführt werden können, was sie nützlicher und profitabler macht. Agilere Quadrokopter, z.B. solche die kleiner gebaut sind, sind jedoch aufgrund deren schnellerer Dynamik auch schwieriger zu regeln. Zudem werden aerodynamische Effekte, die während agilen Manövern mit hohen Geschwindikeiten und Beschleunigungen auftreten, für die Flugcharakteristik von Quadrokoptern relevant. Diese sind jedoch nur schwierig zu modellieren und in die Regelung zu integrieren. Werden diese Effekte vernachlässigt, wird die Flug-Performance nahe dem Schwebeflug zwar nur gering, bei schnellen Flügen allerdings signifikant verschlechtert.

Diese Dissertation ist auf Regelungsmethoden, die es Quadrokoptern erlauben schnellen Trajektorien genau zu folgen, fokussiert. Die präsentierten Regelungsmethoden beinhalten Methoden für die low-level Regelung von Orientierung und Rotationsgeschwindigkeiten sowie für die high-level Positionsregelung. Diese verbessern die Fähigkeit von Quadrokoptern Trajektorien genau zu folgen indem sie die Dynamik der einzelnen Motoren sowie lineare “rotor drag” Effekte berücksichtigen. Des Weiteren beinhaltet diese Dissertation Algorithmen die es Quadrokoptern erlauben sicher aber nahe ihrer physikalischen Grenzen zu operieren und sich von fehlgeschlagenen Manövern zu retten. Zudem beinhaltet sie Beiträge zur Anwendung von Quadrokoptern innerhalb von Roboterteams die für Such- und Rettungsmissionen eingesetzt werden oder enge, schiefe Öffnungen traversieren müssen. Diese Dissertation beinhaltet folgende Beiträge:

- Entwicklung von Quadrokopterinfrastruktur für Forschung im Bereich kamerabasierter Drohnen mit Schwerpunkt auf Regelungsalgorithmen.
- Eine Methode um kamerabasierte Quadrokopter aus beliebigen Lagen mit beliebigen Geschwindigkeiten zu Restabilisieren.
- Der erste Beweis dass die Dynamik eines Quadrokopters unter linearen "rotor drag" Effekten "differentially flat" ist. Ich entwickelte einen Positionsregler der mithilfe dieser Eigenschaft "feed-forward" Regelterme berechnet, die ein genaues folgen einer Trajektorie mit hohen Geschwindigkeiten ermöglichen.
- Eine neue Regelungsmethode für die Orientierung und Rotationsgeschwindigkeiten, die ein genaueres Folgen von Referenzwerten durch die Berücksichtigung der Motordynamik ermöglicht.
- Eine priorisierende Saturierungsmethode die einen stabilen Flug von Quadrokoptern, die nahe an ihren physikalischen Grenzen operieren wo sie Saturierungen ihrere Motoren erreichen, ermöglicht.
- Das ganze Regel-Framework für Quadrokopter, das während dieser Arbeit entwickelt wurde, ist als Open-Source Software erhältlich.
- Ein Algorithmus der die effiziente Zusammenarbeit eines Quadrokopters mit einem Bodenroboter im Rahmen einer Such- und Rettungsmission ermöglicht.
- Einen "Pose"-Schätzer der mithilfe einer Kamera und infrarot LEDs die relative Lokalisierung zwischen Mitglieder eines Roboterteams ermöglicht. Dieser ist als Open-Source Software erhältlich.
- Die erste Methode die es einem Quadrokopter erlaubt mit einem agilen Manöver nur mittels onboard Sensoren und Berechnungen durch enge, schiefe Öffnungen zu fliegen.

List of Contributions

Google Scholar Profile: https://goo.gl/V8MYgb

Journal Publications

- **Matthias Faessler**, Antonio Franchi, and Davide Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robotics and Automation Letters (RA-L)* (2018). DOI: 10.1109/LRA.2017.2776353
  Links: PDF, Video 1, Video 2, Software

- **Matthias Faessler**, Davide Falanga, and Davide Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robotics and Automation Letters (RA-L)* 2.2 (2017), pp. 476–482. DOI: 10.1109/LRA.2016.2640362
  Links: PDF, Video

- **Matthias Faessler**, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor MAV". In: *Journal of Field Robotics* 33.4 (2016), pp. 431–450. DOI: 10.1002/rob.21581
  Links: PDF, Video 1, Video 2, Video 3, Video 4, Software

- Alessandro Giusti, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, **Matthias Faessler**, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots". In: *IEEE Robotics and Automation Letters (RA-L)* 1.2 (2016), pp. 661–667. DOI: 10.1109/LRA.2015.2509024
  Links: PDF, Video

Peer-Reviewed Conference Papers

- Davide Falanga, Elias Mueggler, **Matthias Faessler**, and Davide Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE International Conference on Robotics and Automation (ICRA)* 2017. DOI: 10.1109/ICRA.2017.7989679
  Links: PDF, Video

- **Matthias Faessler**, Flavio Fontana, Christian Forster, and Davide Scaramuzza. "Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor". In: *IEEE International Conference on Robotics and Automation (ICRA)* 2015, pp. 1722–1729. DOI: 10.1109/ICRA.2015.7139420
  Links: PDF, Video

- Christian Forster, **Matthias Faessler**, Flavio Fontana, Manuel Werlberger, and Davide Scaramuzza. "Continuous On-Board Monocular-Vision-based Elevation Mapping Applied to Autonomous Landing of Micro Aerial Vehicles". In: *IEEE International Conference on Robotics and Automation (ICRA)* 2015, pp. 111–118. DOI: 10.1109/ICRA.2015.7138988
  Links: PDF, Video

- Elias Mueggler, **Matthias Faessler**, Flavio Fontana, and Davide Scaramuzza. "Aerial-guided Navigation of a Ground Robot among Movable Obstacles". In: *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)* 2014, pp. 1–8. DOI: 10.1109/SSRR.2014.7017662
  Links: PDF, Video 1, Video 2

- **Matthias Faessler**, Elias Mueggler, Karl Schwabe, and Davide Scaramuzza. "A Monocular Pose Estimation System based on Infrared LEDs". In: *IEEE International Conference on Robotics and Automation (ICRA)* 2014, pp. 907–913. DOI: 10.1109/ICRA.2014.6906962
  Links: PDF, Video, Software

Technical Reports

- **Matthias Faessler**, Antonio Franchi, and Davide Scaramuzza. "Detailed Derivations of 'Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories' ". In: *ArXiv e-prints, arXiv:1712.02402* (2017). Links: PDF, Video

Open-Source Software

- RPG quadrotor control framework.
  - https://github.com/uzh-rpg/rpg_quadrotor_control
  - https://github.com/uzh-rpg/rpg_quadrotor_common
- Single board computer GPIO and ADC library.
  - https://github.com/uzh-rpg/rpg_single_board_io
- RPG monocular pose estimator.
  - https://github.com/uzh-rpg/rpg_monocular_pose_estimator

Awards

- Winner of KUKA Innovation Award (€20.000), 2014 (Video)
- SSRR Best Paper Award Finalist, 2014

Contents

1 Introduction

This thesis presents control algorithms for quadrotors with a focus on enabling agile maneuvers. It describes the quadrotor system developed as a research platform for agile autonomous vision-based flight at the *Robotics and Perception Group*. As a main focus of this thesis, I developed control algorithms that enable quadrotors to accurately track high-speed trajectories. These algorithms consist of a high-level control loop that considers linear rotor drag effects, a low-level control loop that considers the dynamics of the motors, a thrust mixing scheme that considers a varying drag torque of each rotor, and a prioritizing saturation that enables stable flight of quadrotors despite of reaching motor saturations. Additionally, based on the developed quadrotor system, this thesis presents algorithms for relative localization of a quadrotor and a ground robot, the collaboration of a quadrotor and a ground robot among movable obstacles, as well as for letting a quadrotor safely pass a narrow gap.

This thesis is split into three parts: first, it provides details about the developed quadrotor-research platforms at the *Robotics and Perception Group* with a focus on their control. Second, it presents research applications of vision-based quadrotors, mainly for multi-robot collaboration of a quadrotor and a ground robot. Third, it shows methods I developed for accurate tracking of high-speed trajectories with quadrotors by considering rotor-drag effects, motor dynamics, and motor saturations in the controller.

This thesis is structured in the form of a collection of papers. An introductory section that highlights the concepts and ideas behind the thesis is followed by self-contained publications in the appendix.

In the next section, the working principle, applications, advantages, and challenges of quadrotors are discussed. Section 1.2 summarizes the state of the art in quadrotor control and Section 1.3 motivates and states the research objectives of this dissertation. The papers in the appendix are summarized in Chapter 2. Finally, Chapter 3 provides future research directions.

1.1 Quadrotors in Robotics

Over the past decade, quadrotors have gained a tremendous popularity in the robotics community, among hobbyists, and in the consumer market. They are aerial platforms with a simple mechanical design that can be assembled from cheap components suitable for a wide range of applications. Quadrotors are typically built from four motors with fixed-pitch propellers arranged in a cross configuration where all the motor axes are parallel to each other and neighboring propellers spin in opposite directions. The four generated rotor thrusts can be controlled individually by changing the speed of the respective motor. This enables the quadrotor to produce a collective thrust perpendicular to the rotors plane and torques along the three body axes. The four control inputs that control each motor speed are enough to stabilize the six degrees of freedom of a quadrotor since only four of them are independent.

This section summarizes the applications of quadrotors as well as advantages and challenges in the usage of quadrotors and which challenges are tackled by the presented work.

1.1.1 Applications of Quadrotors

Ever since quadrotors and drones in general became affordable and reliable about a decade ago, their number of applications has increased very fast and by now covers essentially every task that involves moving, e.g., sensors or payloads to locations that are hard to reach or even unreachable for humans or ground vehicles. The biggest drone markets currently are inspection and agriculture.[1] Drones are successfully deployed for inspections of power lines, bridges, pipelines (e.g. [128]), railways, and many other infrastructures. In agriculture, among others, drones are used for soil and field analysis, crop monitoring, planting, and crop spraying (e.g. [32]). Besides those, drones are used to support search and rescue missions where they are used for mapping and disaster analysis or even for searching victims [120]. Similarly, drones are used for security and surveillance (e.g. [130], [129]), where they are deployed for monitoring events, the environment (e.g. [109]), general infrastructure, or warehouses. Additionally to these commercial applications, drones are widely used among hobbyists e.g. for aerial photography, which has also become a big commercial market, as well as for fun disciplines, such as first-person-view racing. In the work presented here, we incorporated many of the recent and tremendously fast developments of hardware and software for first-person-view racing from the past years. In the future, drones are expected to be used extensively for package delivery, for which, already today, a lot of investments and research is being done. Furthermore, drones are expected to be deployed for construction tasks (e.g. [4], [73]).

[1] http://uk.businessinsider.com/commercial-uav-market-analysis-2017-8?r=US&IR=T

In summary, most of the applications of quadrotors are for passive tasks, i.e. for moving a sensor to desired locations in unconstrained 3D space. Such tasks are hard and expensive to accomplish without drones but are also what quadrotors are particularly well suited for. Currently, there are many applications for quadrotors and, all together, they form a multi-billion-dollar market, which is expected to grow much more in the coming years and, hence, there is a huge need to exploit the potential of autonomous quadrotors further. To make quadrotors even more useful and more profitable it is crucial to increase their level of autonomy and the speed at which they can operate, since this reduces the number of required operators and makes their task completion faster. Enabling autonomous quadrotors to move faster and more agile is the main focus of this work.

1.1.2 Advantages

One of the main advantages of quadrotors over other types of small scale aerial vehicles, and, with that, the reason for their popularity, is their simple and robust mechanical design. They are built from a rigid, cross-shaped frame, which can be designed lightweight but very robust against crashes. Compared to other multirotors, where all the rotor planes are parallel, quadrotors use the minimum number of motors to control their four independent degrees of freedom and are, therefore, the simplest configuration of such multirotor types. The motors with propellers are the only moving parts of the vehicle. Since these propellers mostly have a fixed blade pitch, their thrust can be controlled by changing the motor speed, which is done by electronic speed controllers (ESCs). Hence, this setup does not require complex mechanical parts, such as a swash plate used on helicopters. Furthermore, the required components to build a quadrotor can be purchased off the shelf in a wide variety and are relatively cheap. This makes them especially suitable for research purposes since they are cheap and offer great flexibility in terms of size and weight, which is required to use new sensors and setups as it is often the case in research experiments.

Besides their simple and robust design, quadrotors have favorable flight capabilities compared to other types of aerial vehicles. Quadrotors can take-off and land vertically and, therefore, only require minimal space for start and landing. Also, they can stay airborne without the requirement to move, which is required in many tasks, such as aerial inspection, surveillance, or photography. Additionally to static flight, quadrotors can be designed to have a rigid structure and a very high thrust-to-weight-ratio enabling them to perform agile maneuvers.

In summary, quadrotors are well suited for research on aerial robotics since they are simple and cheap to design, robust against crashes, can vertically takeoff and land, and yet are very agile.

1.1.3 Challenges

Quadrotors are unstable systems, i.e., stable hover flight cannot be achieved by applying a constant input to all four motors. For stabilizing a quadrotor, its motor inputs have to be controlled at a sufficiently high frequency, which depends on its dynamical characteristics. The smaller a quadrotor, the more agile it gets, i.e., as derived in [119], the maximum angular acceleration scales inversely proportional to the quadrotor size. This might be exploited for agile flight but also requires higher control frequencies to cope with the faster dynamics of the vehicle. Furthermore, the smaller a quadrotor is, the less efficient it gets due to the smaller rotor disk area, which is decreasing quadratically with the quadrotor size. For these reasons, it is difficult to find the optimal configuration of a quadrotor for a certain task since it is a trade off between achievable flight time, agility, and the sensors and computers that can be carried.

Quadrotors are under-actuated systems since they have four control inputs to control their six degrees of freedom. This is not a problem for free flight since only four independent degrees of freedom need to be controlled. But it makes quadrotors not suitable for interactions with their environment since they are not able to exert forces independently of their orientation. This can lead to problems when, e.g., taking off from an inclined surface and should be considered in the take-off procedure [166]. Therefore, for interactions with the environment, aerial vehicles that can exert forces in more than a single axis are typically used, such as helicopters [83] or hexarotors with tilted propellers [144]. Furthermore, when performing agile flights close to the physical limits of a quadrotor, motors might saturate, which leads to an inability to control the four independent degrees of freedom. Therefore, motor saturations can cause a quadrotor to become unstable. This work proposes a solution to achieve stable flight with quadrotors despite of motor saturations.

For agile flight with quadrotors, the biggest challenge is that the relevant dynamics get very complex to describe for maneuvers with hight speeds and accelerations. While aerodynamic effects can be neglected during hover flight, they become relevant in agile maneuvers where certain effects are virtually impossible to be modeled accurately, such as when the rotors enter a so called vortex ring state. Due to this complex modeling which is required to capture the dynamics of a quadrotor in high-speed flight, it is also difficult to consider the full model in control algorithms, which leads to control errors. One goal of this thesis is to incorporate the main aerodynamic effects into the control algorithm to reduce the resulting control errors during agile maneuvers.

This thesis is tackling the following challenges around quadrotors:

- Design of control algorithms that stabilize quadrotors considering that they are under-actuated systems.
- Extending existing quadrotor control methods to consider aerodynamic effects

and motor dynamics to improve trajectory tracking for high-speed trajectories.

- Achieving stable flight when operating quadrotors at their physical limits where their motors saturate.

1.2 State of the Art on Quadrotor Control

This section summarizes the state of the art in quadrotor control and research on aerodynamics of quadrotors. It also indicates how the work of this thesis relates to and extends the state-of-the-art methods.

1.2.1 Quadrotor Systems

Systems for research on quadrotor control and state estimation for drones were developed in many research groups around the world. Especially the systems in [96] and [110] designed for research on quadrotor control served as baseline and inspiration for the development of the quadrotor system at the *Robotics and Perception Group*, which was part of this work. The developed system builds up on these two works that aimed at flying quadrotors in motion capture systems and were adapted and extended to our specific needs. The system presented in [60] puts a lot of emphasis in flexibility of using quadrotors with different interfaces for user inputs, different control methods, and different state estimates. This work also influenced our system architecture which aims at using quadrotors with either state estimates from a motion capture system or from visual odometry algorithms.

1.2.2 Quadrotor Control Methods

The most common control architecture for quadrotors consists of two control loops for position and attitude. Such an architecture is, e.g., found in all the quadrotor systems mentioned above. The high level position control loop typically implements a PD control law on position and velocity with feed-forward terms to compensate for gravity and accelerations from a reference trajectory. The output of this control loop is a desired acceleration which encodes the desired collective thrust and the desired attitude up to a heading which can be chosen independently. The desired attitude is then controlled by a low-level control loop based on the assumption that the attitude dynamics of a quadrotor are much faster than its position and velocity dynamics. A survey of attitude representations and their suitability for attitude control of a rigid body is presented in [28]. In state-of-the-art quadrotor control, the attitude is represented either as a unit quaternion or a rotation matrix. Based on these representations, almost globally asymptotically stable attitude controllers can be designed. Quadrotor control methods consisting of a high-level PD position control loop and a low-level PD attitude control

loop based on rotation matrices are presented in [106] and [89]. Attitude controllers with similar properties based on unit quaternions are presented in [52] and [11]. Since for all these methods asymptotical stability of the high-level position controller and the low-level attitude controller can be shown, also the interconnected system with both loops is asymptotically stable (c.f. [80] Lemma 5.6).

Often, attitude control cannot be achieved by a single control loop due to the commonly used hardware with multiple processing units. Typically, two processing units are embedded on quadrotors, where an onboard computer does state estimation and high-level control and a less powerful but real time micro controller is used for low-level control. Since attitude control requires an estimate of the attitude, which is only available on the onboard computer, it has to run on the onboard computer. To achieve accurate body-rate control, fast and low latency inputs to the motors are required, which can only be achieved by a real time micro controller. Therefore, the attitude control of a quadrotor is often achieved by two cascaded control loops, e.g. in [86], and [19] where the first loop controls the attitude at a low frequency and the second loop controls the body rates at a high frequency. This is not preferable over having a single control loop but often necessary due to the used hardware setup. Due to the used hardware for this work, attitude control can also only be achieved by two cascaded control loops.

In [106], it was shown that the common model of a quadrotor *without* considering rotor drag effects is differentially flat when choosing its position and heading as flat outputs. Furthermore, this work presented a control algorithm that computes the desired collective thrust and torque inputs from the measured position, velocity, orientation, and body-rates errors. With this method, agile maneuvers with speeds of several meters per second were achieved. In [42], the differential flatness property of a hexarotor that takes the desired collective thrust and its desired orientation as inputs was exploited to compute feed-forward terms used in an LQR feedback controller. The desired orientation was then controlled by a separate low-level control loop, which also enables the execution of flight maneuvers with speeds of several meters per second. The differential flatness property was proved and used for control also for many different aerial vehicle configurations such as under actuated aerial vehicles with any number of different parallel manipulator arms attached to their center of mass [179]. Similarly, we extend these works by showing that the dynamics of a quadrotor are differentially flat even when they are subject to linear rotor drag effects. Similarly to [42], we make use of this property to compute feed-forward terms that are then applied by a position controller.

Since quadrotors are under-actuated systems, motor saturations or motor failures can cause them to become unstable when these cases are not considered properly. In [116] it is demonstrated that a quadrotor's position can be stabilized despite the complete loss of up to three propellers. To additionally stabilize the orientation in

case of rotor failures, vehicles with more propellers need to be considered, such as hexarotors and potentially tilted propellers which are well suited as safe platforms with rotor redundancy [112]. In [113] a partial control allocation method that prioritizes the application of desired body torques over collective thrust is used to handle infeasible inputs before applying the motor commands. In this work, I present a saturation scheme that prioritizes control inputs according to their importance for trajectory tracking in case of motor saturations. The presented saturation scheme enables stable quadrotor flight despite motor saturations while being able to make use of the full input range of the individual motors.

Besides the most common control schemes composed of cascaded loops for position and attitude, several other control methods to control a quadrotor have been proposed in the literature. Among these methods are linear quadratic regulators (LQR), model predictive controllers (MPC), and optimal control techniques. LQR based controllers can be implemented as full state feedback controllers for both position and attitude control [138]. Additionally, LQR methods were shown to be suitable for controlling multi body systems such as quadrotors carrying a slung load [30] or balancing an inverted pendulum [63] and [20]. In [77], a fast model predictive attitude controller is proposed, that, together with a high-level linear quadratic regulator achieves tracking of aggressive trajectories. Model predictive controllers were also used as high-level controllers with a separate attitude control loop in e.g. [137], [117] and [9]. Furthermore, fast trajectory generation methods can be exploited in a model predictive control fashion by only applying the first command resulting from a planned trajectory and then replan a new trajectory for the next control loop iteration [66]. Optimal control methods were used in [140] and [67] for finding time optimal trajectories to transition between two states. The structure of a transition maneuver is obtained by Pontryagin's minimum principle and then a numerical algorithm is proposed to solve the boundary value problem induced by the minimum principle to compute maneuvers for arbitrary initial and final states. Based on a two-dimensional first-principles quadrotor model, this method finds trajectories that are bang-bang in the thrust command, and bang-singular in the body rate control. An iterative optimal control algorithm that finds both a trajectory and a stabilizing controller simultaneously is proposed in [30]. The capabilities of this algorithm are demonstrated by flying a quadrotor with slung load through a window not high enough for the load to pass while hanging straight down as well as by go-to-goal tasks with single and double rotor failures.

1.2.3 Accurate Trajectory Tracking

Following trajectories with a quadrotor accurately becomes increasingly more difficult with increasing speed due to modeling errors that become more relevant at higher speeds and accelerations. One way to overcome this limitation is by repeating a given trajectory and iteratively learn how to control inputs need to be adapted to improve

trajectory tracking accuracy. In [95], open-loop multi flips and time-optimal translation maneuvers were learned by repeating them and adapting trajectory parameters. In [64], an iterative learning scheme in the frequency domain for learning periodic maneuvers with quadrotors is presented. The proposed learning scheme takes the control errors as inputs and outputs a setpoint shift that improves trajectory tracking accuracy. Furthermore, a time scaling method allows the transfer of learnt maneuvers to different execution speeds through a prediction of the disturbance change. The achieved tracking errors after learning a figure-eight trajectory were similar to the deviations between different runs of the trajectory. This approach was also applied for flying fast circles with a quadrotor while balancing an inverted pendulum in [65]. As opposed to learning input correction, in [149] the motion parameters of a periodic trajectory are adjusted to improve temporal and spatial tracking. The motion parameters are either estimated online or offline prior to flight to avoid initial transients. Similarly, [148] estimates the disturbances due to modeling errors along a desired trajectory using a Kalman filter. Based on the estimated disturbances, a more adequate input for the next trial is computed by solving a constrained optimization problem. To overcome the problem of not generalizing to different trajectories, [61] shows that the major dynamics of the iterative learning control process can be captured by a linear map, which can then be used to improve the initialization of learning unseen trajectories. A different approach in learning to improve trajectory tracking for unseen trajectories is taken in [91] and [180] where a deep neural network is used as an add-on block to a classical feedback controller. The deep neural network is trained offline on recorded data from a set of training trajectories and achieves a significant reduction of the trajectory tracking error.

In contrast to learning control inputs that compensate for modeling errors, considering a more accurate dynamical model in the controller should lead to similar results. Improving the dynamical model has the advantage that once the model parameters are identified, any trajectory can be flown more accurately without iteratively learning it. This is the approach I followed in this thesis by incorporating the motor dynamics and aerodynamic drag effects into the dynamical model of a quadrotor which is used for control design. In this work, for improving body-rate control, I considered the motor dynamics in the low-level controller, which was also done in related work. In [178], cascaded PID controllers are designed and enhanced with Smith predictors to incorporate the dynamics of the motors for full quadrotor attitude control on SO(3). An LQR attitude controller for a single axis, which is extended with first order dynamics of the motors is presented in [78] and [171]. In contrast, I propose an LQR controller for the coupled 3D body rates, incorporating the motor dynamics, which also provides feedback linearization and feed forward on desired angular accelerations. Incorporating aerodynamic drag effects into the dynamical model of a quadrotor is discussed in the following section.

1.2.4 Quadrotor Aerodynamics

Thanks to the well established theory of full size helicopters, the influence of aerodynamic effects on quadrotor dynamics were considered since the beginning of quadrotor research [134], [68]. Since then, modeling of these effects for quadrotors was intensely studied starting from the well established theory of full sized helicopters. In [12], very detailed models of aerodynamic forces of rotors used on small scale quadrotors were derived using momentum and blade element theory. This work also studied the influence of different blade geometries commonly used on quadrotors. Blade flapping and thrust variation due to translational speed was modeled in [71] based on momentum theory. Simple methods to improve control performance were proposed by compensating for the resulting moment due to blade flapping and the thrust variation due to translation. In [22] and [102], blade flapping was modeled with blade element theory showing that the main resulting force is linear in the translational speed. These works point out the influence of the vertical location of a quadrotor's center of gravity on its stability as well as implications on controller feedback. Additionally to investigating quadrotor aerodynamics in vertical and forward flight, [135] also investigates proximity effects such as ground and wall effects and effects from neighboring vehicles. In [150] and [57], a tethered quadrotor is used to identify aerodynamic parameters such as linear and quadratic drag coefficients as well as thrust variations due to speed. Thanks to the tether, the quadrotor reaches very high velocities on a horizontal circle despite the limited available space. An estimation framework for identifying model parameters of a multi rotor vehicle including aerodynamic parameters is proposed in [24]. This work shows that the quadratic body drag is negligible below speeds of about $5\,\mathrm{m\,s^{-1}}$ and becomes relevant above in spite of being smaller than the linear rotor drag effects during the conducted experiments in a wind tunnel with maximum wind speeds of $15\,\mathrm{m\,s^{-1}}$. The dynamical model of a quadrotor used in this thesis is based on these works and especially makes use of the model presented in [76] including linear rotor drag effects, for which we show differential flatness. These rotor drag effects originate from blade flapping and induced drag of the rotors, which are, as suggested in [100], combined as linear effects in a lumped parameter dynamical model thanks to their equivalent mathematical expression.

Already the works in [22], [102], and [12] suggested that the dynamical models of quadrotors incorporating linear drag effects can be used for velocity estimation, where [22] and [102] show this experimentally. In [90], a quadrotor model including linear drag effects is used for improving state estimation with an extended Kalman filter. This work shows that the attitude and velocity estimation can be improved even when only using an inertial measurement unit and no position measurements. Similarly, [25] proposed a model based state estimation scheme for micro aerial vehicles based on an extended Kalman filter. By only using measurements from an inertial measurement unit and a barometric pressure sensor, this estimation scheme keeps the estimated velocity bounded, which is crucial when using this state estimate as

a backup if no position estimate is available. Additionally to estimating horizontal velocities, [5] extends estimation to full body-fixed-frame velocity measurement by exploiting their previous work in aerodynamic modeling of rotor performance and measurements of mechanical power supplied to the rotor hub. This work proposes a nonlinear observer to estimate the attitude and body-fixed linear velocity jointly using an inertial measurement unit and electronic speed controllers that measure the rotor speed and torque.

There exist several approaches to incorporate aerodynamic effects into the control design of quadrotors. In [10], accurate thrust control is achieved by electronic speed controllers through a model of the aerodynamic power generated by a fixed-pitch rotor under wind disturbances, which reduces the trajectory tracking error of a quadrotor. However, this method requires specific hardware and long-lasting calibration compared to other state-of-the-art algorithms [51] that achieve thrust control through fast and accurate speed control of the motors. Rotor drag effects were considered in control methods for multi-rotor vehicles in [76] and [127], where the control problem was simplified by decomposing the rotor drag force into a component that is independent of the vehicle's orientation and one along the thrust direction, which leads to an explicit expression for the desired thrust direction. While [76] models the rotor drag to be proportional to the square root of the thrust, which is proportional to the rotor speed and physically the correct relation, [127] models the rotor drag to be proportional to the thrust. In [161], a refined thrust model and a control scheme that considers rotor drag in the computation of the thrust command and the desired orientation are presented. However, this scheme does not use feed-forward terms on body rates and angular accelerations, which does not allow perfect trajectory tracking. Additionally to the thrust command and desired orientation, the control scheme in [8] also computes the desired body rates and angular accelerations by considering rotor drag but requires estimates of the quadrotor's acceleration and jerk, which are usually not available. In this work, I prove that the dynamical model of a quadrotor subject to linear rotor drag effects as developed in [76] is differentially flat in its position and heading. This property is then used to compute the exact reference thrust, orientation, body rates, and angular accelerations directly from a reference trajectory to be tracked, which are then used as feed-forward terms in the controller. This results in a comparably simple and intuitive control law that theoretically enables perfect tracking of high-speed trajectories.

1.3 Research Objectives

This section summarizes the research objectives of this thesis. The objectives consist of, from scratch, setting up a quadrotor platform capable of agile flight, which will then be used for developing control algorithms that enable accurate tracking of high-speed trajectories.

Vision-Based Quadrotor Research Platform. When this work started, there existed no common infrastructure for conducting experiments with quadrotors at the *Robotics and Perception Group*. Therefore, the first objective of this work was to set up a quadrotor system to serve as a standard platform for experiments on new control and vision-based state estimation algorithms. This system has to be flexible enough to connect new sensors used for state estimation but also has to provide low level access to provide full control over the applied control algorithms. Since this work focuses on developing new control algorithms for quadrotors, one goal of building up a quadrotor research platform is to create a complete custom control architecture to provide access at every level of control. Furthermore, to be easily used as a platform for other experiments, the control method should allow to use a quadrotor also without detailed knowledge about it.

Accurate Tracking of High-Speed Trajectories. The main objective of this work is to improve tracking accuracy of quadrotors executing high-speed trajectories. This tackles the entire control method from a low-level controller that can accurately apply desired high-level control inputs to a high-level controller that considers all relevant dynamics during high-speed flight. Such dynamics are mostly aerodynamic effects, which only become relevant at high speeds and, therefore, need to be investigated and integrated into the high-level control method. An emphasis is put in applying accurate feed-forward control to the lowest necessary level, which is crucial for achieving accurate trajectory tracking.

Multi-Robot Collaboration. Apart from developing quadrotor control algorithms, this work also investigates how quadrotors can be applied in a robot team, where they collaborate with ground robots. This requires the development of a relative localization strategy between the quadrotor and the ground robot. Furthermore, it should investigate how such a team of robots can be deployed most effectively by exploiting the individual strengths and capabilities of the robots.

2 Contributions

This chapter summarizes the key contributions of the papers that are reprinted in the appendix. It further highlights the connections between the individual results and refers to related video and open-source code contributions. In total, this research has been published in five peer-reviewed conference publications and four journal publications. One further journal paper is currently under review at the *IEEE Transactions on Robotics (TRO)*. These works led to the *KUKA Innovation Award 2014*, which included prize money of 20,000 euros. They were also successfully demonstrated live hundreds of times in multiple countries around the world and served as basis for experiments of about half of the papers published by the *Robotics and Perception Group*. The control algorithms developed within this work are available as open-source code.[1]

2.1 Quadrotor Control System

Since the *Robotics and Perception Group* was founded in the same year as the presented works started, the entire infrastructure for autonomous, vision-based quadrotor flight had to be set up first. For this, I co-developed a quadrotor system based on off-the-shelf components around a PixHawk flight controller [105] with custom software for high- and low-level control. On these quadrotor platforms, the visual-odometry pipeline SVO [50] was integrated, which was also developed in house, for achieving autonomous quadrotor flight with only on-board sensors and on-board computation. To this end, the developed system serves as standard platform for experiments and robot demonstrations by the *Robotics and Perception Group*.

[1] https://github.com/uzh-rpg/rpg_quadrotor_control

2.1.1 Paper A: Infrastructure for Autonomous, Vision-Based Quadrotor Flight

(P1) M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor MAV". in: *J. Field Robot.* 33.4 (2016), pp. 431–450. ISSN: 1556-4967. DOI: 10.1002/rob.21581

Quadrotors have a huge potential for autonomous, mapping, monitoring, and inspection tasks since they can easily overcome obstacles on the ground and reach remote locations that are hard to access or dangerous for humans. Especially vision-based quadrotors are suitable for such tasks since they can fly independently of any external infrastructure or positioning systems such as GPS and are therefore able to e.g. also enter buildings. We developed a vision-based quadrotor system that is capable of fully autonomous flights, i.e. it only relies on onboard sensors (a camera and an inertial measurement unit) and computation. The developed control methods run fully onboard the vehicle and are well integrated with a visual odometry pipeline that was developed at the *Robotics and Perception Group*. Together with a ground station computer, which runs a 3D reconstruction pipeline developed at the *Robotics and Perception Group*, our quadrotor can provide a 3D reconstruction of a specified area from an aerial perspective in real time. Our quadrotors serve as standard robotic platforms at the *Robotics and Perception Group*, which are used for experiments on a daily basis. We also performed hundreds of demonstrations based on this system at our laboratory and in several countries around the world.

Related Software

(S1) http://rpg.ifi.uzh.ch/software_datasets.html

Related Datasets

(D1) http://rpg.ifi.uzh.ch/software_datasets.html

Related Videos

(V1) https://youtu.be/7-kPiWaFYAc

(V2) https://youtu.be/sdu4w8r_fWc

(V3) https://youtu.be/3mNY9-DSUDk

(V4) https://youtu.be/JbACxNfBI30

(V5) https://youtu.be/LssgKdDz5z0

(a) Quadrotor platform **(b)** 3D Reconstruction

Figure 2.1: Autonomous, vision-based quadrotor platform capable of generating 3D reconstructions. (a): A closeup of the developed quadrotor platform: 1) down-looking camera, 2) Odroid U3 quad-core computer, 3) PixHawk autopilot. (b): A 3D reconstruction captured with one of our autonomous quadrotors. The reconstruction is computed in real time while the quadrotor executed the trajectory overlaid in blue.

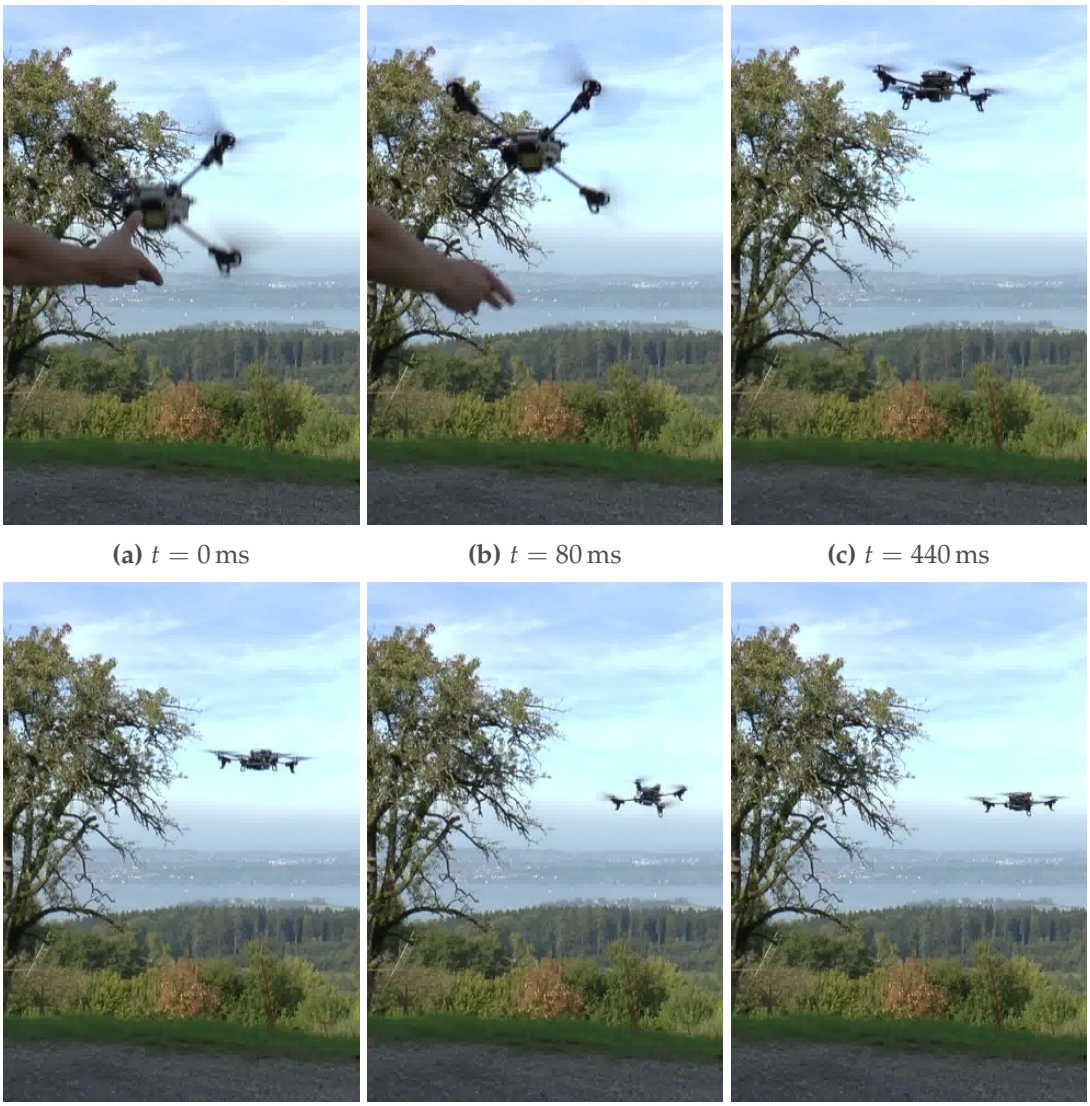
2.1.2 Paper B: Re-Initialization and Failure Recovery

(P2) M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza. "Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 1722–1729. DOI: 10.1109/ICRA.2015.7139420

Autonomous, vision-based quadrotor flight is widely regarded as a challenging perception and control problem since the accuracy of a flight maneuver is strongly influenced by the quality of the on-board state estimate. In addition, any vision-based state estimator can fail due to the lack of visual information in the scene or due to the loss of feature tracking after an aggressive maneuver. We propose a system that enables a monocular-vision–based quadrotor to automatically recover from any unknown, initial attitude with significant velocity, such as after loss of visual tracking due to an aggressive maneuver. We present an almost globally stable attitude controller based on quaternions that is able to bring the quadrotor into an upright position from any initial attitude. After this, the quadrotor re-initializes its vision-based state estimation pipeline to regain fully controlled autonomous flight. This procedure is successfully used for launching a quadrotor by throwing it in the air and for failure recovery during aggressive flight maneuvers. With this, our recovery system is an important milestone towards safe agile flight with quadrotors. The developed high- and low-level controllers are used as standard controllers for the quadrotor system used at the *Robotics and Perception Group*.

Related Videos

(V6) https://youtu.be/pGU1s6Y55JI



(a) $t = 0\,\text{ms}$ **(b)** $t = 80\,\text{ms}$ **(c)** $t = 440\,\text{ms}$

(d) $t = 1{,}120\,\text{ms}$ **(e)** $t = 1{,}640\,\text{ms}$ **(f)** $t = 2{,}000\,\text{ms}$

Figure 2.2: Autonomous recovery after throwing the quadrotor by hand: (a) the quadrotor detects free fall and (b) starts to control its attitude to be horizontal. Once it is horizontal, (c) it first controls its vertical velocity and then, (d) its vertical position. The quadrotor uses its horizontal motion to initialize its visual-inertial state estimation and uses it (e) to first break its horizontal velocity and then (f) lock to the current position.

2.2 Applications of Quadrotor System

In this section, we present applications of our quadrotor platforms for aerial-ground robot collaborations as well as for flying through inclined narrow gaps.

2.2.1 Paper C: Pose Estimation for Collaboration

(P3) M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza. "A Monocular Pose Estimation System based on Infrared LEDs". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 907–913. DOI: 10.1109/ICRA.2014.6906962

For the collaboration of multiple robots, a relative localization among the robots is necessary for operation in a shared space. To provide such a relative localization, we propose a monocular pose estimation system based on infrared LEDs. The LEDs are mounted on one robot and observed by a camera mounted on another robot, which is equipped with an infrared-pass filter. We find the correspondences between LEDs and image detections by a combinatorial approach and then track them with a constant-velocity model. We then compute the pose by first solving a P3P problem and then refine the pose by minimizing the reprojection error in an optimization. Since the system works in the infrared spectrum, it is robust to cluttered environments and illumination changes that systems working in the visible spectrum suffer from. We show that our system outperforms state-of-the-art approaches in a variety of experiments. Furthermore, we successfully apply our system to stabilize a quadrotor while it is being observed by a ground robot. We released the implementation of our monocular pose estimation system as open-source software.

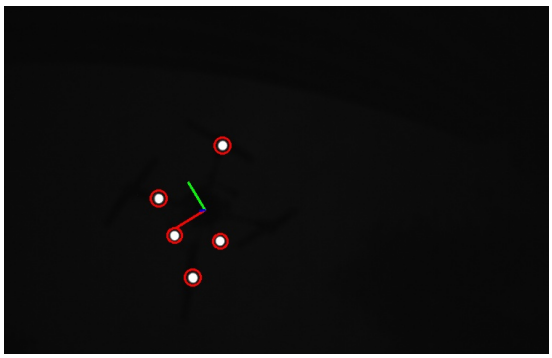
Related Software

(S2) https://github.com/uzh-rpg/rpg_monocular_pose_estimator

Related Videos

(V7) https://youtu.be/8Ui3MoOxcPQ

(a) Stabilizing a quadrotor above a ground robot



(b) View from the camera with an infrared-pass filter on the ground robot

Figure 2.3: A camera with an infrared-pass filter is mounted on a ground robot and used to stabilize a quadrotor above it (a). The red circles in (b) indicate LED detections. The pose estimate is illustrated by the projection of the body-fixed coordinate frame of the quadrotor.

2.2.2 Paper D: Aerial and Ground Robot Collaboration

(P4) E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza. "Aerial-guided Navigation of a Ground Robot among Movable Obstacles". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. 2014, pp. 1–8. DOI: 10.1109/SSRR.2014.7017662

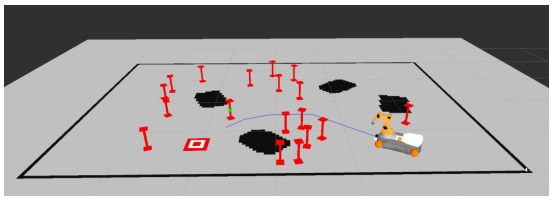
When collaborating, a heterogeneous team of robots can make use of the individual strengths of each robot in the team. Especially for complex tasks where one single robot does not have all the required capabilities such a collaboration is beneficial. We demonstrate the fully autonomous collaboration of an aerial and a ground robot in a mock-up disaster scenario exploiting their individual capabilities. In this scenario, the aerial robot first maps an area of interest, then it computes the fastest mission for the ground robot to reach an identified target location and deliver a package. Such a mission for the ground robot includes driving and removing obstacles in the way while being constantly monitored and commanded by the aerial robot. Our mission-planning algorithm distinguishes between movable and fixed obstacles and considers both the time for driving and removing obstacles. Our system was successfully demonstrated several dozens of times at a trade fair and during demonstrations in our laboratory. By demonstrating this system, we won the *KUKA Innovation Award 2014*, which included prize money of 20.000€.

Related Videos

(V8) https://youtu.be/C5I190lzDdQ

(V9) https://youtu.be/OFPv3BegbFg

(a) Our robots operating in a mock-up disaster site



(b) Corresponding mission plan to guide the ground robot to the target location

Figure 2.4: After mapping the area in (a), the aerial robot is guiding the ground robot to the goal location by a mission plan illustrated in (b). All paths are blocked by obstacles, some of which can be removed by the ground robot.

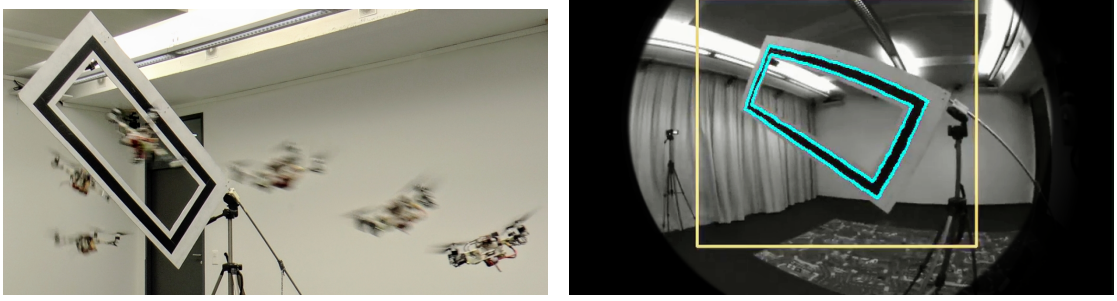
2.2.3 Paper E: Aggressive Flight through Narrow Gaps

(P5) D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017. DOI: 10.1109/icra.2017.7989679

We address one of the biggest challenges towards autonomous quadrotor flight in complex environments, which is flight though narrow gaps. For this, we equipped a quadrotor with a front-looking camera, an inertial measurement unit, and an onboard computer to autonomously detect a gap and traverse it by only using onboard sensing and computing. We estimate the quadrotor's state by computing its relative pose to the gap from the captured images and fuse it with measurements from the inertial measurement unit. We then compute a trajectory that enables the quadrotor to safely pass narrow, inclined gaps with an agile maneuver. Our method generates a trajectory that considers geometric, dynamic, and perception constraints: during the approach maneuver, the quadrotor always faces the gap to allow state estimation, while respecting the vehicle dynamics; during the traverse through the gap, the distance of the quadrotor to the edges of the gap is maximized. We replan the trajectory during its execution to cope with the varying uncertainty of the state estimate. In real experiments, we demonstrate a success rate of 80 % for gap inclinations of up to 45° with out approach.

Related Videos

(V10) https://youtu.be/meSItatXQ7M



(a) The quadrotor passing through the gap **(b)** View from the onboard camera

Figure 2.5: (a) Sequence of our quadrotor passing through a narrow, 45°-inclined gap. Our state estimation fuses gap detection from a single on-board forward-facing camera (b) with an IMU. All planning, sensing, control run fully on-board a smartphone computer.

2.3 Accurate Agile Quadrotor Flight

In this section, we present control algorithms that specifically improve the trajectory tracking accuracy when executing agile maneuvers with quadrotors. With all these works together we achieved a significant improvement in trajectory tracking when flying at velocities of several meters per second.

2.3.1 Paper F: Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag

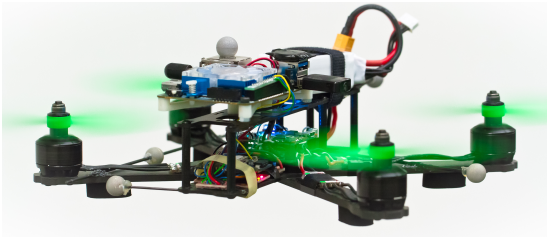
(P6) M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 620–626. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2776353

In most state-of-the-art quadrotor controllers, aerodynamic effects that become significant at velocities of several meters per second are neglected and therefore their trajectory tracking performance suffers at high velocities. To overcome this issue, we propose a high-level quadrotor control method that computes feed-forward terms considering linear rotor drag effects, which allows accurate tracking of agile trajectories. For this we prove that the dynamical model of a quadrotor subject to linear rotor drag effects is differentially flat in its position and heading. With this property, we can compute the reference control inputs that compensate for rotor drag effects as algebraic functions of a reference trajectory. We show that our method reduces the root mean squared tracking error by 50 % compared to state-of-the-art control methods that do not consider rotor drag independently of the executed trajectory. Furthermore, we propose a method based on a gradient-free optimization to identify the rotor drag coefficients which are required to compute the feed-forward control terms.

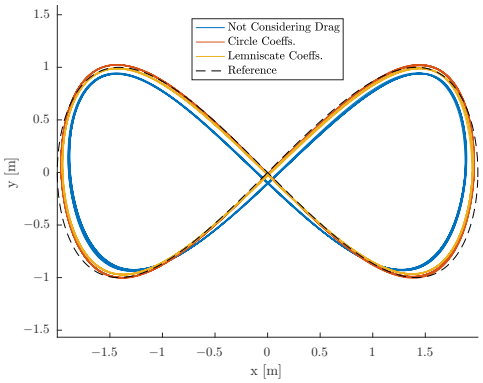
Related Videos

(V11) https://youtu.be/VIQILwcM5PA

(V12) https://youtu.be/LmMgx_vKh5s



(a) First-person-view racing inspired quadrotor platform used for experiments



(b) Trajectory tracking performance on a lemniscate trajectory

Figure 2.6: (a) Quadrotor platform we used for experimentally evaluating trajectory tracking improvement when considering rotor drag effects in the controller. (b) Ground truth position for ten loops on a lemniscate trajectory without considering rotor drag (solid blue), with drag coefficients estimated on a circle trajectory (solid red), and with drag coefficients estimated on the lemniscate trajectory (solid yellow) compared to the reference position (dashed black). The maximum speed on the trajectory is $4.0\,\mathrm{m\,s^{-1}}$.

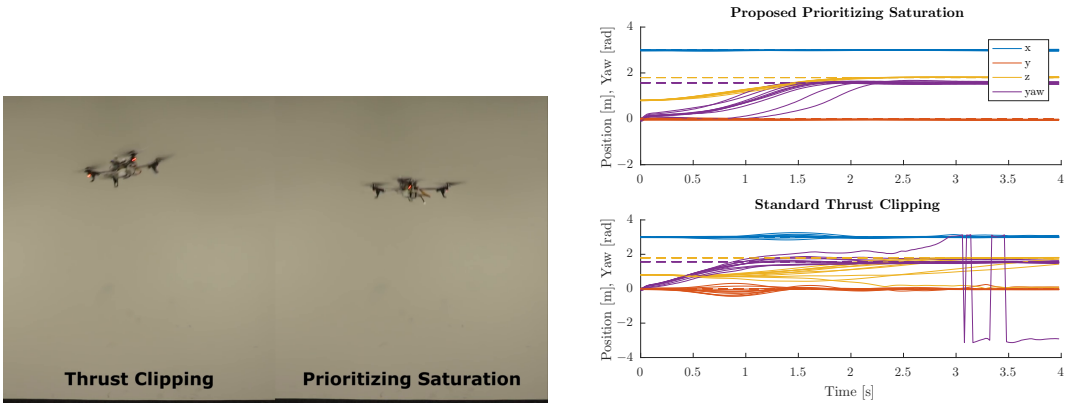
2.3.2 Paper G: Low-Level Control and Motor Saturation

(P7) M. Faessler, D. Falanga, and D. Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 476–482. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2640362

Most control schemes for quadrotors are designed for near hover conditions where they work well. However, executing fast trajectories with such control schemes leads to tracking errors that get larger the faster the quadrotor flies. To accurately track a trajectory, the commands from a high-level position controller must be tracked well. To achieve this, we propose a low-level body-rate controller, an iterative thrust-mixing scheme, and a prioritizing motor-saturation scheme. Our body-rate controller uses LQR-control methods to consider both the body rate and the single motor dynamics, which reduces the overall trajectory-tracking error while still rejecting external disturbances well. Our iterative thrust-mixing scheme computes the four rotor thrusts given the inputs from a position-control pipeline. Through the iterative computation, we are able to consider a varying ratio of thrust and drag torque of a single propeller over its input range, which allows applying the desired yaw torque more precisely and hence reduces the yaw-control error. Our prioritizing motor-saturation scheme improves stability and robustness of a quadrotor's flight and may prevent unstable behavior in case of motor saturations. We successfully demonstrate the improved trajectory tracking, yaw-control, and robustness in case of motor saturations in real-world experiments with a quadrotor.

Related Videos

(V13) https://youtu.be/6YEMxFgToyg



(a) Quadrotor performing a step in height and yaw

(b) Corresponding ground truth data for ten runs

Figure 2.7: (a) Comparison of simple thrust clipping and our prioritizing saturation scheme during a simultaneous step in height and yaw. (b) Response in position and yaw for ten runs where the quadrotor simultaneously performs a 1 m step in height and a 90° step in yaw starting at $t = 0$ s using prioritizing saturation (top) and thrust clipping (bottom). The reference values after the step are: $x = 3.0$ m, $y = 0.0$ m, $z = 1.8$ m, and $yaw = ß/2$ rad. Solid lines show ground truth data and dashed lines show desired reference values.

2.4 Unrelated Contributions

During the Ph.D., two papers were co-authored that are not part of the Ph.D. work itself but made use of the developed quadrotor system. U1 used our quadrotor system as a test platform for real-time 3D terrain reconstruction and landing-spot detection, whereas U2 used it to apply a deep neural network to steer a drone autonomously along forest or mountain trails.

(U1) C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza. "Continuous On-Board Monocular-Vision–based Aerial Elevation Mapping for Quadrotor Landing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 111–118. URL: http://dx.doi.org/10.1109/ICRA.2015.7138988

(U2) A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots". In: *IEEE Robot. Autom. Lett.* 1.2 (July 2016), pp. 661–667. ISSN: 2377-3766. DOI: 10.1109/LRA.2015.2509024

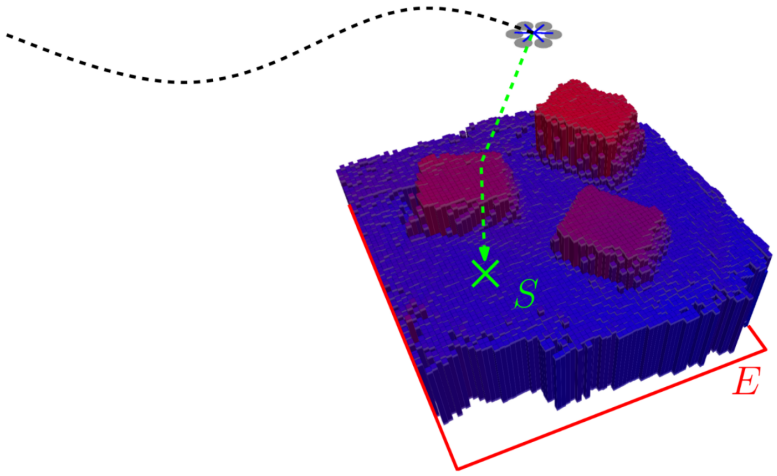


Figure 2.8: U1: Real-time 3D terrain reconstruction and autonomous landing-spot detection.

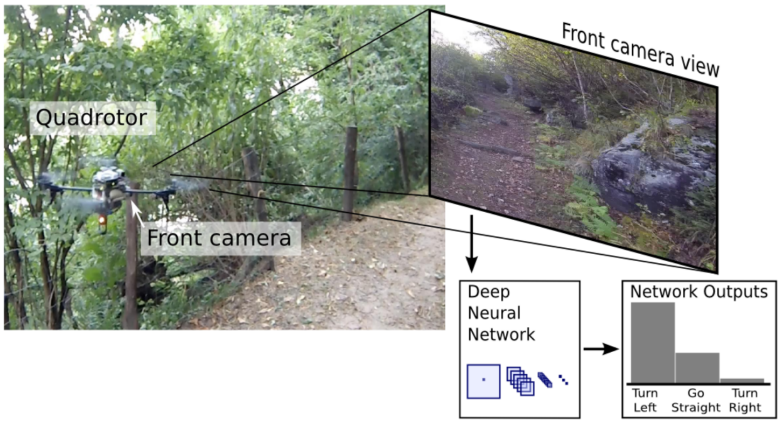


Figure 2.9: U2: Autonomously following forest or hiking trails with a drone steered by a deep neural network.

3 Future Directions

For over a decade, quadrotors have been used widely in research, hobby, and commercial applications but their potential has not been fully exploited yet. Especially the agility of quadrotors, which is one of their biggest advantages, has not been exploited much outside of remote controlled hobby applications. Therefore, especially in research, we have to push the limits of quadrotors more, keep integrating the latest available hardware, and further develop control algorithms that enable exploiting the full potential in terms of agility and autonomy of quadrotors.

Pushing the limits. To this date, flights with quadrotors in research experiments are almost exclusively conducted with low velocities of up to only a few meters per second and accelerations that require a thrust to weight ratio of less than two. Quadrotor platforms can nowadays easily be built with a thrust to weight ratio of five or larger with maximum reachable speeds of forty meters per second and more. Since these limits were extended drastically over the past years through fast hardware development, quadrotors are currently operated far from their limitations. Therefore, it is important for the research community to push control algorithms for quadrotors to a level where they can exploit their full potential especially in terms of agility. To achieve this, we have to reach higher velocities and accelerations in quadrotor experiments to explore and potentially exploit effects that only then become relevant.

New Integrated Hardware. Recent developments of new integrated hardware such as the Qualcomm Snapdragon Flight board[1] bring a number of advantages and possibilities for agile quadrotor flight. Since such a board contains all the required electronics for autonomous quadrotor flight, it allows building more compact platforms which is favorable for agile flight. Furthermore, the physical separation of high- and low-level control that is typically used on quadrotors by a single board computer and a micro controller is no longer necessary. This has the advantage that the state estimate is also available in the low-level controller and, therefore, only two control loops for

[1] https://developer.qualcomm.com/hardware/snapdragon-flight/board-specs

position and attitude are required, where the attitude can also be controlled at a hight frequency in the low level controller. Additionally, running both loops on one device has the advantage to reduce latencies from measurement to control action, which is very important for fast and precise quadrotor control, and these latencies can be considered precisely since only one clock is used for timing both control loops. Finally, such new platforms with new electronic speed controllers enable to receive feedback about the speed of all motors, which can then be incorporated into the control algorithms to enable more precise control of rotor thrusts under various conditions.

Generation of Agile Trajectories. At the moment, the most used type of trajectories for flying quadrotors are polynomials, such as minimum snap trajectories [106]. Polynomial trajectories have two major drawbacks. The first drawback is that there is no guarantee of obtaining trajectories without unnecessary curves, i.e., when designing a polynomial trajectory between two states, the trajectory might temporarily move the quadrotor away from the goal position. The second drawback is that physical limits are difficult to enforce and that they can only be enforced at one point on the trajectory, which might result in operating the quadrotor far from the limits for most of the time on the trajectory. Therefore, new concepts of designing trajectories for agile flight with quadrotors are required. A great demonstrator for this is autonomous drone racing where the full potential in agility of quadrotors should be exploited. This requires trajectory generation methods used for e.g. car racing in 2D [53] to be extended to 3D. While for race cars this typically boils down to combinations of trajectory segments of maximum throttle and maximum friction force [88] with some intermediate segments, this will relate to maximum thrust segments with some intermediate segments. Such trajectories may be obtained through classical optimization techniques [67] or model predictive control strategies. Also, from observing human drone race pilots, such trajectories might contain motion primitives such as fast turns that are executed almost open loop, i.e., without feedback on position, velocity, or attitude. This was demonstrated to be feasible for autonomous quadrotors, e.g., for doing multiple flips [95].

Learning Dynamics. Within this work, we realized that the relevant dynamics of a quadrotor become increasingly more difficult to be captured by a dynamical model with increasing speeds and acceleration during operation. Already now, many assumptions need to be made to use these dynamical models in model based control approaches and we are still operating quadrotors far from their physical limitations. Newly emerging techniques in machine learning might be key to move away from increasingly complicated dynamical models and enable accurate capturing of quadrotor dynamics at high speeds and accelerations. Such techniques could be used to either learn the dynamics of a quadrotor or, as mentioned above, learn motion primitives, such as sharp turns (similarly to learning flips in [95]), which can then be connected to full trajectories. First works in this direction are presented in [91] and [180] where a deep

neural network is used as an add-on block to a classical feedback controller to improve trajectory tracking for unseen trajectories. The deep neural network is trained offline on recorded data from a set of training trajectories to adapt the reference signals to the feedback control loop. Potentially, this is similar to obtaining a more accurate dynamical model as I proposed in this work by incorporating rotor drag effects into the the model and controller. These two approaches reduce the trajectory tracking error by a similar but significant amount and potentially could reduce it even further when being combined.

Summary

The potential of quadrotors in terms of agility is far from being exploited with current platforms and control algorithms. The key towards a full exploitation of agility lies in recent and current hardware design that enables more compact designs with higher computational power, lower latency control loops, and more feedback of the quadrotor's state. This will enable to redesign control algorithms incorporating additional feedback, such as motor speed measurements, and running control loops faster with less latency for more accurate control. It will also enable new control methods, such as control policies obtained through machine learning, that do not rely on complicated dynamical models, which might improve trajectory tracking accuracy at high speeds and accelerations significantly.

A Autonomous, Vision-based Flight and Live Dense 3D Mapping

M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor MAV". in: *J. Field Robot.* 33.4 (2016), pp. 431–450. ISSN: 1556-4967. DOI: 10.1002/rob.21581

Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle

Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza

Abstract — The use of mobile robots in search-and-rescue and disaster-response missions has largely increased over the recent years. However, they are still remotely controlled by expert professionals on an actuator set-point level and would, therefore, benefit from any bit of autonomy added. This would allow them to execute high-level commands, such as "*execute this trajectory*" or "*map this area*". In this paper, we describe a vision-based quadrotor Micro Aerial Vehicle (MAV) that can autonomously execute a given trajectory and provide a live, dense three-dimensional (3D) map of an area. This map is presented to the operator while the quadrotor is mapping, so that there are no unnecessary delays in the mission. Our system does not rely on any external positioning system (e.g., GPS or motion capture systems) as sensing, computation, and control are performed fully onboard a smartphone processor. Since we use standard, off-the-shelf components from the hobbyist and smartphone markets, the total cost of our system is very low. Due to its low weight (below 450 g), it is also passively safe and can be deployed close to humans. We describe both the hardware and the software architecture of our system. We detail our visual odometry pipeline, the state estimation and control, and our live dense 3D mapping, with an overview of how all the modules work and how they have been integrated into the final system. We report the results of our experiments both indoors and outdoors. Our quadrotor was demonstrated over 100 times at multiple trade fairs, at public events, and to rescue professionals. We discuss the practical challenges and lessons learned. Code, datasets,

and videos are publicly available to the robotics community.

Supplementary Material

This paper is accompanied by videos demonstrating the capabilities of our platform in outdoor and indoor scenarios:

- Indoor evaluation (disturbance and autonomous, vision-based live 3D mapping): http://youtu.be/sdu4w8r_fWc
- Outdoor autonomous, vision-based flight over disaster zone: http://youtu.be/3mNY9-DSUDk
- Outdoor autonomous, vision-based flight with live 3D mapping: http://youtu.be/JbACxNfBI30

More videos can be found on our Youtube channel: https://www.youtube.com/user/ailabRPG/videos

Our visual odometry code (called SVO) for vision-based navigation has been released open source and is freely available on the authors' homepage.

A.1 Introduction

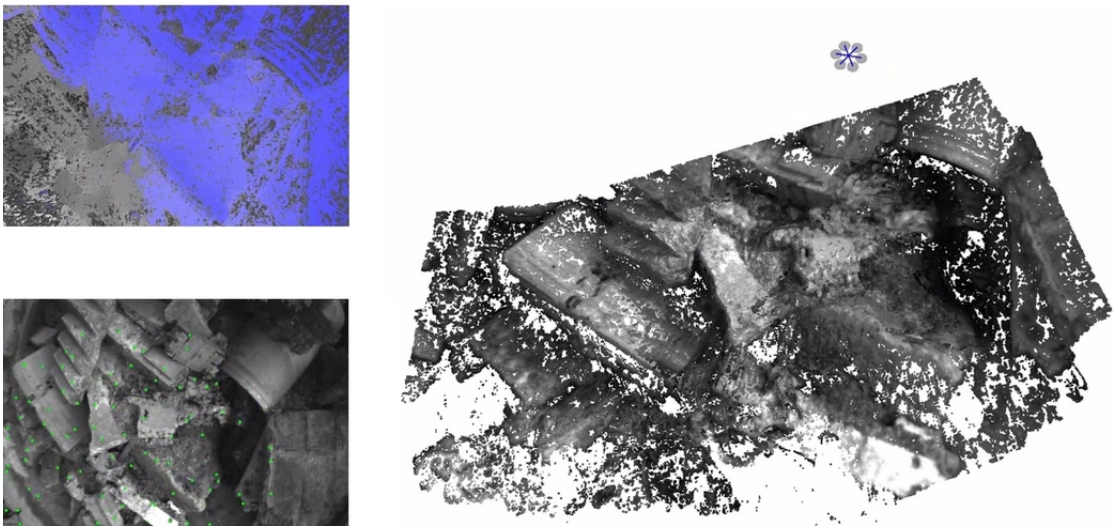
A.1.1 Motivation

For search-and-rescue, disaster response, and surveillance missions, it is crucial for human rescuers to get an overview of the situation in order to take appropriate measures. In this paper, we present a vision-based quadrotor that can autonomously execute a trajectory, build a dense 3D map of an unknown area in real-time, and present it live to a user during the mission. A live feedback is, indeed, crucial to avoid any unnecessary delays during a rescue operation.

When robots are deployed in disaster situations, three expert professionals are on average required for each robot to control them [120]. Additionally, they are teleoperated on an actuator set-point level, which makes the execution of tasks slow and cumbersome.

At the current state, all Micro Aerial Vehicles (MAVs) used in search-and-rescue and remote-inspection scenarios are controlled under direct line of sight with the operator [120]. If wireless communication can be maintained, there is the possibility

(a) Our quadrotor mapping a mock-up disaster site



(b) Live dense 3D map

Figure A.1: Our system (a) can autonomously build a dense 3D map of an unknown area. The map is presented to the operator through a graphical user interface on the base station laptop while the quadrotor is mapping (b, right inset) together with the onboard image (b, bottom left inset) and a confidence map (b, top left inset).

to teleoperate the MAV by transmitting video streams from the onboard cameras to the operator. However, teleoperation from video streams is extremely challenging in indoor environments.

Since such systems exhibit very limited or no autonomy at all, the stress level on the operator is very high, which limits the mission time drastically. Operator errors could harm the robot or, even worse, cause further damage. For these reasons, there is a large need of flying robots that can navigate autonomously, without any user intervention, or execute high-level commands, such as "*execute this trajectory*" or "*map this area*". This would bring several advantages over today's disaster-response robots. Firstly, the robot could easily be operated by a single person, who could focus on the actual mission. Secondly, a single person could operate multiple robots at the same time to speed up the mission. Finally, rescuers could operate such systems with very little training. These advantages, in combination with the low-cost of the platform, will soon make MAVs become standard tools in disaster response operations.

A.1.2 System Overview

Our system consists of a quadrotor and a laptop base station with a graphical user interface for the operator (see Figure A.1). The quadrotor is equipped with a single, down-looking camera, an inertial measurement unit (IMU), and a single-board computer. All required sensing, computation, and control is performed onboard the quadrotor. This design allows us to operate safely even when we temporarily lose wireless connection to the base station. It also allows us to operate the quadrotor beyond the range of the wireless link as, for instance, inside buildings. We do not require any external infrastructure such as GPS, which can be unreliable in urban areas or completely unavailable indoors.

We chose quadrotors because of their high maneuverability, their ability to hover on a spot, and their simple mechanical design. To make the system self-contained, we rely only on onboard sensors (i.e., a camera and an IMU).

For operating in areas close to humans, safety is a major concern. We aim at achieving this passively by making the quadrotor as lightweight as possible (below 450 g). Therefore, we chose passive sensors, which are typically lighter and consume less power. However, when using cameras, high computational power is required to process the huge amount of data. Due to the boost of computational power in mobile devices (e.g., smartphones, tablets), high-performance processors that are optimized for power consumption, size, and cost are available today. An additional factor for real-world applications is the cost of the overall system. The simple mechanical design and the use of sensors and processors produced millionfold for smartphones makes the overall platform low cost (1,000 USD).

A.1.3 Contributions and Differences with other Systems

The main contribution of this paper is a self-contained, robust, low-cost, power-on-and-go quadrotor MAV that can autonomously execute a given trajectory and provide a live dense 3D mapping without any user intervention and using only a single camera and an IMU as the main sensory modality.

The most similar to our system is the one which resulted from the European project SFLY [174, 145]. However, in SFLY, dense 3D maps were computed offline and were available only after several minutes after landing.

Another difference with the SFLY project lies in the vision-based motion-estimation pipeline. Most monocular, visual-odometry algorithms used for MAV navigation (see Section A.2.2) rely on PTAM [81], which is a feature-based visual SLAM algorithm, running at 30 Hz, designed for augmented reality applications in small desktop scenes. In contrast, our quadrotor relies on a novel visual odometry algorithm (called SVO, which we proposed in [49]) designed specifically for MAV applications. SVO eliminates the need of costly feature extraction and matching as it operates directly on pixel intensities. This results in high precision, robustness, and higher frame-rates (at least twice that of PTAM) than current state-of-the-art methods. Additionally, it uses a probabilistic mapping method to model outliers and feature-depth uncertainties, which provide robustness in scenes with repetitive, and high-frequency textures.

In SFLY, a commercially-available platform was used, with limited access to the low-level controller. Conversely, we have full access to all the control loops, which allows us to tune the controllers down to the lowest level. Furthermore, we propose a one-time-per-mission estimation of sensor biases, which allows us to reduce the number of states in the state estimator. In addition, the estimated biases are also considered in the low-level controller (body rate controller), while in SFLY, they were only used in the high-level controller (position controller). Furthermore, we propose a calibration of the actually-produced thrust, which ensures the same control performance regardless of environmental conditions such as temperature and air pressure.

A.1.4 Outline

The paper is organized as follows. Section A.2 reviews the related work on autonomous MAV navigation and real-time 3D dense reconstruction. Section A.3 presents the hardware and software architecture of our platform. Section A.4 describes our visual odometry pipeline, while Section A.5 details the state estimation and control of the quadrotor. Section A.6 describes our live, dense 3D reconstruction pipeline. Finally, Section A.7 presents and discusses the experimental results and Section A.8 comments on the lessons learned.

A.2 Related Work

To date, most autonomous MAVs rely on GPS to navigate outdoors. But, GPS is not reliable in urban settings and is completely unavailable indoors. Because of this, most works on autonomous indoor navigation of flying robots have used external motion-capture systems (e.g., Vicon or OptiTrack). These systems are very appropriate for testing and evaluation purposes [110, 96], such as prototyping control strategies or executing fast maneuvers. However, they need pre-installation of the cameras and, thus, cannot be used in unknown, yet unexplored environments. Therefore, for truly autonomous navigation in indoor environments, the only viable solution is to use onboard sensors. The literature on autonomous navigation of MAVs using onboard sensors includes range (e.g., laser rangefinders or RGB-D sensors) and vision sensors.

A.2.1 Navigation based on Range Sensors

Laser rangefinders have been largely explored for Simultaneous Localization and Mapping (SLAM) with ground mobile robots [165]. Because of the heavy weight of 3D scanners (see for instance the Velodyne sensor (more than 1 kg)), laser rangefinders currently used on MAVs are only 2D. Since 2D scanners can only detect objects that intersect their sensing plane, they have been used for MAVs in environments characterized by vertical structures [6, 2, 153, 154] and less in more complex scenes.

RGB-D sensors are based upon structured-light techniques, and, thus, share many properties with stereo cameras. However, the primary differences lie in the range and spatial density of depth data. Since RGB-D sensors illuminate a scene with a structured-light pattern, contrary to stereo cameras, they can estimate depth in areas with poor visual texture but are range-limited by their projectors. RGB-D sensors for state estimation and mapping with MAVs have been used in [7], as well as in [154, 152], where a multi-floor autonomous exploration and mapping strategy was presented.

A.2.2 Navigation based on Vision Sensors

Although laser rangefinders and RGB-D sensors are very accurate and robust to illumination changes, they are too heavy and consume too much power for lightweight MAVs. In this case, the alternative solution is to use vision sensors. Early works on vision-based navigation of MAVs focused on biologically-inspired algorithms (like optical flow) to perform basic maneuvers, such as take-off and landing, reactive obstacle avoidance, corridor and terrain following [143, 69, 182, 181, 93]. Since optical flow can only measure the relative velocity of image features, the position estimate of the MAV will inevitably drift over time. This can be avoided using visual odometry or visual SLAM methods, in monocular [173, 174, 49, 145] or stereo configurations [2, 155, 147]. Preliminary experiments for MAV localization using a visual EKF-based

SLAM technique were described in [3]. However, the first use of visual SLAM to enable autonomous basic maneuvers was done within the framework of the SFLY European project, where a single camera and an IMU were used for state estimation and point-to-point navigation over several hundred meters in an outdoor, GPS-denied environment [174, 145].

Most monocular Visual Odometry (VO) algorithms for MAVs [15, 34, 174, 145] rely on PTAM [81]. PTAM is a feature-based SLAM algorithm that achieves robustness through tracking and mapping several hundreds of features. Additionally, it runs in real-time (at around 30 Hz) by parallelizing the motion estimation and mapping tasks and by relying on efficient keyframe-based Bundle Adjustment (BA) [158]. However, PTAM was designed for augmented reality applications in small desktop scenes and multiple modifications (e.g., limiting the number of keyframes) were necessary to allow operation in large-scale outdoor environments [174].

A.2.3 Real-time Monocular Dense 3D Mapping

In robotics, a dense reconstruction is needed to interact with the environment—as in obstacle avoidance, path planning and manipulation. Moreover, the robot must be aware of the uncertainty affecting the measurements in order to intervene by changing the vantage point or deploying different sensing modalities [48].

A single moving camera represents the most general setting for stereo vision. Indeed, in stereo settings a fixed baseline constrains the operating range of the system, while a single moving camera can be seen as a stereo camera with an adjustable baseline that can be dynamically re-configured according to the requirements of the task.

Few relevant works have addressed real-time, dense reconstruction from a single moving camera and they shed light on some important aspects. If, on one hand, estimating the depth independently for every pixel leads to efficient, parallel implementations, on the other hand the authors of [55, 159, 125, 175] argued that, similar to other computer vision problems, such as image de-noising [142] and optical flow estimation [176], a smoothing step is required in order to deal with noise and spurious measurements. In [159], smoothness priors were enforced over the reconstructed scene by minimizing a regularized energy functional based on aggregating a photometric cost over different depth hypothesis and penalizing non-smooth surfaces. The authors showed that the integration of multiple images leads to significantly higher robustness to noise. A similar argument is put forth in [125], where the advantage of photometric cost aggregation [162] over a large number of images taken from nearby viewpoints is demonstrated.

However, despite the ground-breaking results, these approaches present some limitations when addressing tasks in robot perception. Equally weighting measurements

Component	Weight (g)	Price (USD)
Frame, Gears, Propellers	119	63
Motors, Motor Controllers	70	214
PX4FMU, PX4IOAR	35	181
Hardkernel Odroid-U3	49	65
Camera, Lens	16	326
1,350 mA h Battery	99	44
Other Parts	54	110
Total	**442**	**1003**

Table A.1: Weight and price of the individual components of our quadrotors.

from small and large baselines, in close and far scenes, causes the aggregated cost to frequently present multiple or no minima. Depending on the depth range and sampling, these failures are not always recoverable by the subsequent optimization step. Furthermore, an inadequate number of images can lead to a poorly constrained initialization for the optimization and erroneous measurements that are hard to detect. It is not clear how many images should be collected, depending on the motion of the camera and the scene structure. Finally, the number of depth hypotheses controls the computational complexity, and the applicability is, thus, limited to scenes bounded in depth.

Therefore, in [133] we presented the REMODE framework that overcomes these limitations by using a probabilistic approach handling measurement uncertainty. We build on the Bayesian depth estimation proposed in [170] for per-pixel depth estimation and introduce an optimization step to enforce spatial regularity over the recovered depth map. We propose a regularization term based on the weighted Huber norm but, differently from [125], we use the depth uncertainty to drive the smoothing and exploit a convex formulation for which a highly parallelizable solution scheme has been recently introduced [27].

A.3 System Overview

We propose a system consisting of a quadrotor equipped with a monocular camera and a laptop serving as ground station. The quadrotor is able to navigate fully autonomously without requiring any communication with the ground station. On the ground station, we can compute a dense 3D reconstruction from the images taken by the quadrotor in real time. In the following, we describe the aerial platform and the software modules.

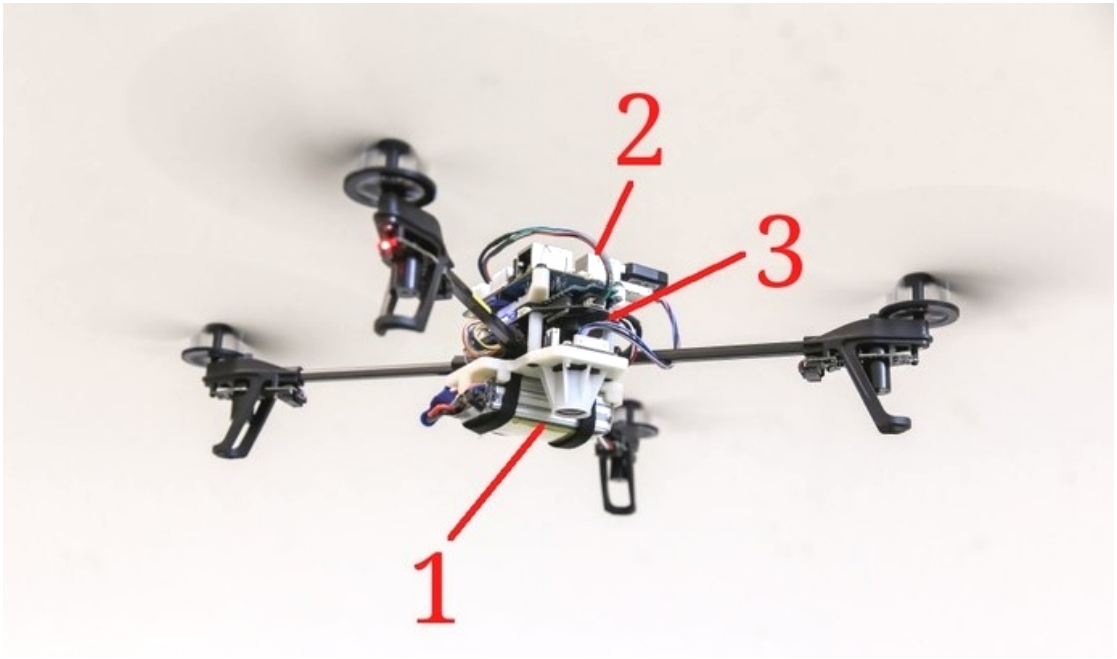


Figure A.2: A closeup of our quadrotor: 1) down-looking camera, 2) Odroid U3 quad-core computer, 3) PIXHAWK autopilot.

A.3.1 Aerial Platform

We built our quadrotor from selected off-the-shelf components and custom 3D printed parts (see Figure A.2). The components were chosen according to their performance and their ability to be easily customized.

Our quadrotor relies on the frame of the Parrot AR.Drone 2.0[1] including their motors, motor controllers, gears, and propellers. To reduce play and vibrations on the platform, we replaced the bushings of the propeller axes by ball bearings. The platform is powered by one 1,350 mA h LiPo battery which allows a flight time of 10 min.

We completely replaced the electronic parts of the AR.Drone by a PX4FMU autopilot and a PX4IOAR adapter board developed in the PIXHAWK Project [105]. The PX4FMU consists, among other things, of an IMU and a micro controller to read the sensors, run a body rate controller, and command the motors. Additionally to the PX4 autopilot, our quadrotors are equipped with an Odroid-U3 single-board computer.[2] It contains a 1.7 GHz quad-core processor running XUbuntu 13.10[3] and ROS.[4] The PX4 micro controller communicates with the Odroid board over UART, whereas the Odroid board communicates with the ground station over 5 GHz WiFi.

[1] http://ardrone2.parrot.com/
[2] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G138745696275
[3] http://www.xubuntu.org/
[4] http://www.ros.org/

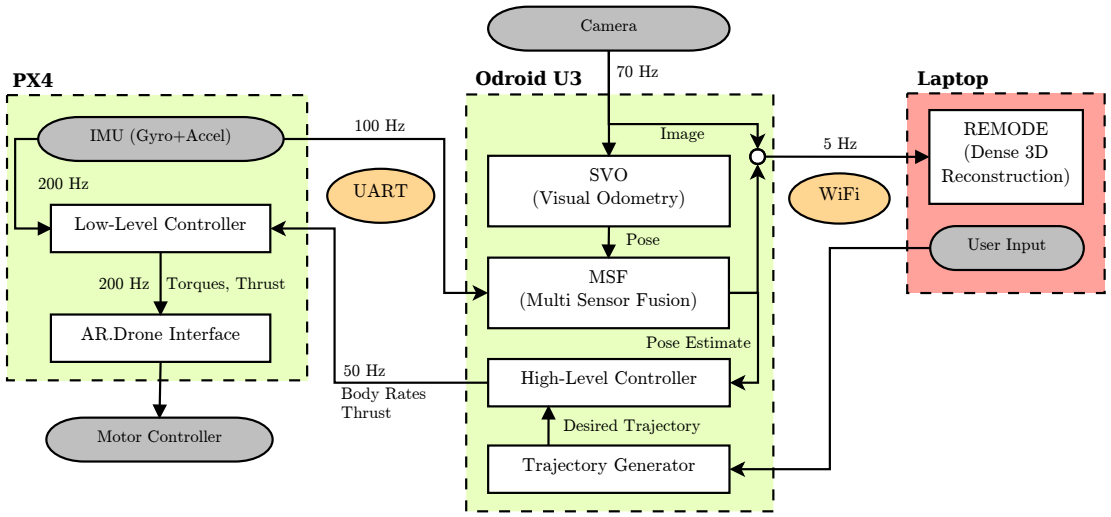


Figure A.3: System overview: the PX4 and Odroid U3 communicate with each other over a UART interface. Communication to the Laptop is over WiFi. Gray boxes are sensors and actuators, software modules are depectied as white boxes.

To stabilize the quadrotor, we make use of the gyros and accelerometers of the IMU on the PX4FMU as well as a downward-looking MatrixVision mvBlueFOX-MLC200w 752×480-pixel monochrome camera with a 130-degree field-of-view lens.[5]

Our platform is easily reparable due to off-the-shelf components, inexpensive (1,000 USD, c.f. Table A.1), lightweight (below 450 g), and, due to its flexible propellers, safe to use.

A.3.2 Software Modules

The software used in our system runs on three different processing units (see Figure A.3), namely the PX4 micro controller, the Odroid computer, and a laptop, which serves as ground station. All the computations required to stabilize the quadrotor are performed onboard. On the ground station (a W530 Lenovo laptop), only the dense 3D reconstruction is computed using its Graphics Processing Unit (GPU).

The PX4 micro controller reads the IMU and controls the desired body rates and collective thrust that it receives from the high-level controller running on the Odroid.

The Odroid processes the camera images by means of our Semi-direct visual odometry (SVO [49]) pipeline (see Section A.4). The visual odometry pipeline outputs an unscaled pose which is then fused with the IMU readings in an Extended Kalman Filter framework (Multi Sensor Fusion (MSF) [98]) to compute a metric state estimate. From this state estimate and a reference trajectory, we compute the desired body rates and

[5] http://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html

collective thrust, which are then sent to the PX4. Alongside this pipeline, we send the images from the camera together with the corresponding pose estimate to the ground station.

On the ground station, we run a dense 3D reconstruction [133] in real time using the camera images with their corresponding pose estimate (see Section A.6).

A.4 Semi-Direct Visual Odometry *(SVO)*

Using Visual Odometry with a single downward-looking camera, we can simultaneously estimate the motion of the vehicle (up to an unknown scale and rigid-body transformation) and the local structure of the scene in the form of a sparse point-cloud. In [49], we proposed a novel VO algorithm called *SVO* that is two times faster in terms of processing time compared to previous methods. The motivation to increase the frame-rate is twofold: first, it allows the MAV to fly faster and more agilely; second, as we will show in the experimental results, it allows the MAV to localize in environments of highly repetitive and high-frequency texture (see Figure A.11).

Figure A.4 provides an overview of SVO. The algorithm uses two parallel threads, one for estimating the camera motion with respect to the local map, and a second one for extending the map as the environment is being explored. This separation allows fast and constant-time tracking in one thread, while the second thread extends the map, decoupled from hard real-time constraints. In the following, we provide an overview of both motion estimation and mapping. We refer the reader to [49] for more details.

A.4.1 Motion Estimation

Methods that estimate the camera pose with respect to a map (i.e., a set of key-frames and 3D points) can be divided into two classes:

(A) Feature-based methods extract a sparse set of salient image features in every image; match them in successive frames using invariant feature descriptors; finally, recover both camera motion and structure using epipolar geometry [146]. The bottleneck of this approach is that approximately 50% of the processing time is spent on feature extraction and matching, which is the reason why most of the VO algorithms still run at 20-30 fps despite the availability and advantages of high frame-rate cameras.

(B) Direct methods [72], on the other hand, estimate the motion directly from intensity values in the image. The rigid-body transformation is found through minimizing the photometric difference between corresponding pixels, where the local intensity gradient magnitude and direction is used in the optimization. Since this approach starts directly with an optimization, increasing the frame-rate means that the optimization is

initialized closer to the optimum, and, thus, it converges much faster. Hence, direct methods in general benefit from increasing the frame-rate [62] as the processing time per frame decreases.

In [49] we proposed a *Semi-Direct Visual Odometry* (SVO) that combines the benefits of feature-based and direct methods. SVO establishes feature-correspondence by means of direct methods rather than feature extraction and matching. The advantage is increased speed (up to 70 fps onboard the MAV and 400 Hz on a consumer laptop) due to the lack of feature-extraction at every frame and increased accuracy through subpixel feature correspondence. Once feature correspondences are established, the algorithm continues using only point-features; hence, the name "*semi-direct*". This switch allows us to rely on fast and established frameworks for bundle adjustment, i.e., joint optimization of both 3D points and camera poses [168].

A.4.2 Mapping

The mapping thread estimates the depth at new 2D feature positions (we used FAST corners [141]) by means of a *depth filter*. Once the depth filter has converged, a new 3D point is inserted in the map at the found depth and immediately used for motion estimation. The same depth filter formulation is used for dense reconstruction and its formulation is explained in more detail in Section A.6.1.

New depth-filters are initialized whenever a new keyframe is selected in regions of the image where few 3D-to-2D correspondences are found. A keyframe is selected when the distance between the current frame and the previous keyframe exceeds 12% of the average scene depth. The filters are initialized with a large uncertainty in depth and with a mean at the current average scene depth.

At every subsequent frame, the epipolar line segment in the new image corresponding to the current depth confidence interval is searched for an image patch that has highest correlation with the reference feature patch (see Figure A.6). If a good match is found, the depth filter is updated in a recursive Bayesian fashion (see Equation (A.31)). Hence, we use many measurements to verify the position of a 3D point, which results in considerably less outliers compared to triangulation from two views.

A.4.3 Implementation Details

The source-code of SVO is available open-source at `github.com/uzh-rpg/rpg_svo`. No special modifications are required to run SVO onboard the MAV. For all experiments, we use the *fast* parameter setting that is available on-line. The *fast* setting limits the number of features per frame to 120, maintains always at maximum 10 keyframes in the map (i.e., older keyframes are removed from the map) and for processing-time

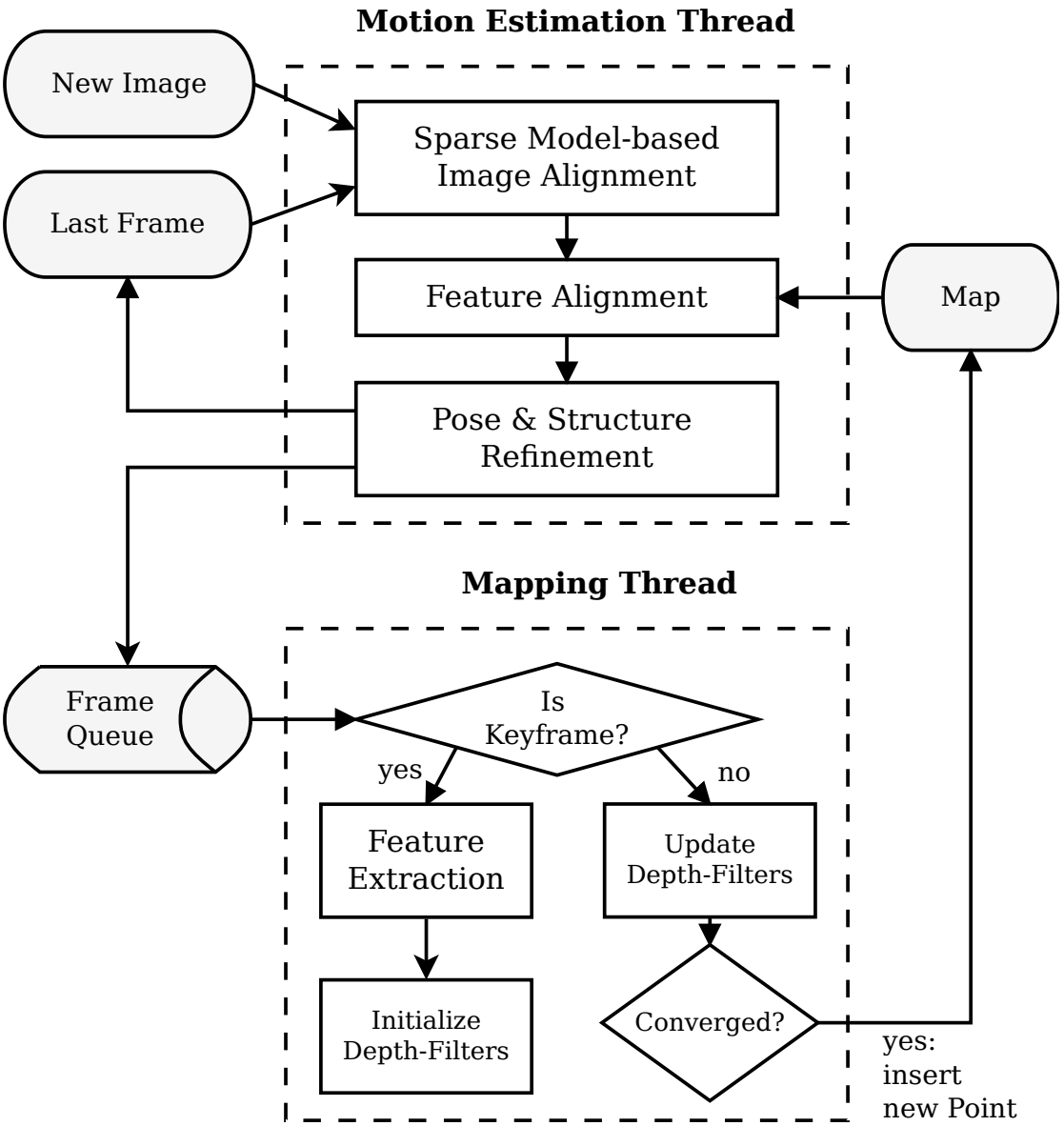


Figure A.4: SVO system overview. Two concurrent threads are used, one for estimating the motion of the camera w.r.t. the map and the second one for extending the map.

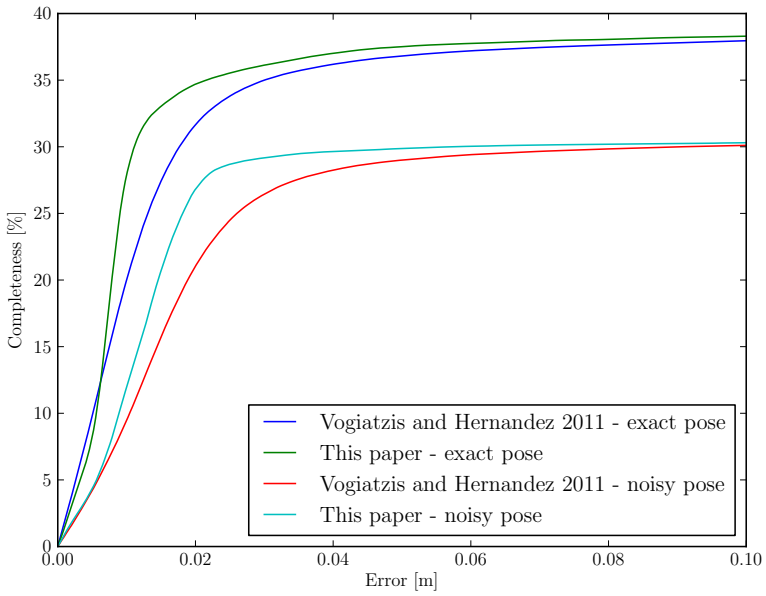


Figure A.5: Quadrotor with coordinate system and rotor forces.

reasons no bundle adjustment is performed.

A.5 State Estimation and Control

In this section, we describe the state estimation and control used to stabilize our quadrotor. Furthermore, we explain how we estimate sensor biases and the actually-produced thrust. The control and calibration sections are inspired by [96].

A.5.1 Dynamical Model

For state estimation and control we make use of the following dynamical model of our quadrotor:

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{A.1}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \mathbf{R} \cdot \mathbf{c}, \tag{A.2}$$

$$\dot{\mathbf{R}} = \mathbf{R} \cdot \hat{\omega}, \tag{A.3}$$

$$\dot{\omega} = \mathbf{J}^{-1} \cdot (\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}), \tag{A.4}$$

where $\mathbf{r} = [x \ y \ z]^T$ and $\mathbf{v} = [v_x \ v_y \ v_z]^T$ are the position and velocity in world coordinates, $\mathbf{R}$ is the orientation of the quadrotor's body coordinates with respect to the world coordinates, and $\boldsymbol{\omega} = [p \ q \ r]^T$ denotes the body rates expressed in body

Parameter	Description	Value	Unit
J_{xx}	x-Axis Moment of Inertia	0.001	$\mathrm{kg\,m^2}$
J_{yy}	y-Axis Moment of Inertia	0.001	$\mathrm{kg\,m^2}$
J_{zz}	z-Axis Moment of Inertia	0.002	$\mathrm{kg\,m^2}$
m	Quadrotor Mass	0.45	kg
l	Quadrotor Arm Length	0.178	m
κ	Rotor Torque Coefficient	0.0195	m
p_{xy}	Horizontal Position Control Gain	5.0	$\mathrm{s^{-2}}$
p_z	Vertical Position Control Gain	15.0	$\mathrm{s^{-2}}$
d_{xy}	Horizontal Velocity Control Gain	4.0	$\mathrm{s^{-1}}$
d_z	Vertical Velocity Control Gain	6.0	$\mathrm{s^{-1}}$
p_{rp}	Roll/Pitch Attitude Control Gain	16.6	$\mathrm{s^{-1}}$
p_{yaw}	Yaw Attitude Control Gain	5	$\mathrm{s^{-1}}$
p_{pq}	Roll/Pitch Rate Control Gain	33.3	$\mathrm{s^{-1}}$
p_r	Yaw Rate Control Gain	6.7	$\mathrm{s^{-1}}$

Table A.2: Parameters and control gains used for the experiments.

coordinates. The skew symmetric matrix $\hat{\boldsymbol{\omega}}$ is defined as

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}. \tag{A.5}$$

We define the gravity vector as $\mathbf{g} = [0 \;\; 0 \;\; -g]^T$ and $\mathbf{J} = \mathrm{diag}\left(J_{xx},\, J_{yy},\, J_{zz}\right)$ is the second-order moment-of-inertia matrix of the quadrotor. The mass-normalized thrust vector is $\mathbf{c} = [0\; 0\; c]^T$, with

$$mc = f_1 + f_2 + f_3 + f_4, \tag{A.6}$$

where f_i are the four motor thrusts as illustrated in Figure A.5. The torque inputs $\boldsymbol{\tau}$ are composed of the single-rotor thrusts as

$$\boldsymbol{\tau} = \begin{bmatrix} \frac{\sqrt{2}}{2} l (f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} l (-f_1 - f_2 + f_3 + f_4) \\ \kappa (f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \tag{A.7}$$

where l is the quadrotor arm length and κ is the rotor-torque coefficient.

The used coordinate systems and rotor numbering are illustrated in Figure A.5, the used parameter values are listed in Table A.2.

A.5.2 State Estimation

To stabilize our quadrotor, we need an estimate of the metric pose as well as the linear and angular velocities. We compute this state estimate by fusing the sensor data from the IMU and the output of the visual odometry in an Extended Kalman Filter. To do so, we make use of an open-source multi sensor fusion package [98]. Since we did not modify this package, we are not describing the sensor fusion in more detail here.

A.5.3 Controller

To follow reference trajectories and stabilize the quadrotor, we use cascaded controllers. The high-level controller running on the Odroid includes a position controller and an attitude controller. The low-level controller on the PX4 contains a body rate controller. The used control gains are listed in Table A.2.

High-Level Control

The high-level controller takes a reference trajectory as input and computes desired body rates that are sent to the low-level controller. A reference trajectory consists of a reference position $\mathbf{r}_{ref}$, a reference velocity $\mathbf{v}_{ref}$, a reference acceleration $\mathbf{a}_{ref}$, and a reference yaw angle ψ_{ref}. First, the position controller is described followed by the attitude controller. The two high-level control loops are synchronized and run at 50 Hz.

Position Controller To track a reference trajectory, we implemented a PD controller with feed-forward terms on velocity and acceleration:

$$\mathbf{a}_{des} = \mathbf{P}_{pos} \cdot \left(\mathbf{r}_{ref} - \hat{\mathbf{r}}\right) + \mathbf{D}_{pos} \cdot \left(\mathbf{v}_{ref} - \hat{\mathbf{v}}\right) + \mathbf{a}_{ref}, \tag{A.8}$$

with gain matrices $\mathbf{P}_{pos} = \text{diag}\left(p_{xy}, p_{xy}, p_z\right)$ and $\mathbf{D}_{pos} = \text{diag}\left(d_{xy}, d_{xy}, d_z\right)$. Since a quadrotor can only accelerate in its body z direction, $\mathbf{a}_{des}$ enforces two degrees of the desired attitude. Now we want to compute the desired normalized thrust such that the z component of $\mathbf{a}_{des}$ is reached with the current orientation. To do so, we make use of the last row of (A.2) to compute the required normalized thrust c_{des} as

$$c_{des} = \frac{\mathbf{a}_{des,z} + g}{\mathbf{R}_{3,3}}. \tag{A.9}$$

The output of the position controller is composed of the desired accelerations $\mathbf{a}_{des}$, which, together with the reference yaw angle ψ_{ref}, encodes the desired orientation as well as a mass normalized thrust c_{des}.

Attitude Controller In the previous paragraph, we computed the desired thrust such that the desired acceleration in the vertical world direction is met. Since a quadrotor can only produce thrust in the body z direction, the attitude controller has to rotate the quadrotor in order to achieve the desired accelerations in the world x-y plane with the given thrust. The translational behavior of the quadrotor is independent of a rotation around the body z axis. Therefore, we first discuss the attitude controller for roll and pitch and, second, we present the yaw controller.

For the x-y plane movement, we restate the first two rows of (A.2) and insert the normalized thrust from (A.9)

$$\begin{bmatrix} \mathbf{a}_{des,x} \\ \mathbf{a}_{des,y} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{1,3} \\ \mathbf{R}_{2,3} \end{bmatrix} c_{des}, \tag{A.10}$$

where $\mathbf{R}_{i,j}$ denotes the (i,j) element of the orientation matrix $\mathbf{R}$. Solving for the two entries of $\mathbf{R}$ which define the x-y plane movement, we find

$$\begin{bmatrix} \mathbf{R}_{des,1,3} \\ \mathbf{R}_{des,2,3} \end{bmatrix} = \frac{1}{c_{des}} \begin{bmatrix} \mathbf{a}_{des,x} \\ \mathbf{a}_{des,y} \end{bmatrix}. \tag{A.11}$$

To track these desired components of $\mathbf{R}$, we use a proportional controller on the attitude error.

$$\begin{bmatrix} \dot{\mathbf{R}}_{des,1,3} \\ \dot{\mathbf{R}}_{des,2,3} \end{bmatrix} = p_{rp} \begin{bmatrix} \mathbf{R}_{1,3} - \mathbf{R}_{des,1,3} \\ \mathbf{R}_{2,3} - \mathbf{R}_{des,2,3} \end{bmatrix}, \tag{A.12}$$

where p_{rp} is the controller gain and $\dot{\mathbf{R}}$ is the change of the orientation matrix per time step.

We can write the first two rows of (A.3) as

$$\begin{bmatrix} \dot{\mathbf{R}}_{des,1,3} \\ \dot{\mathbf{R}}_{des,2,3} \end{bmatrix} = \begin{bmatrix} -\mathbf{R}_{1,2} & \mathbf{R}_{1,1} \\ -\mathbf{R}_{2,2} & \mathbf{R}_{2,1} \end{bmatrix} \cdot \begin{bmatrix} p_{des} \\ q_{des} \end{bmatrix}. \tag{A.13}$$

Finally, the desired roll and pitch rate can be computed by plugging (A.12) into (A.13) and solving for p_{des} and q_{des}.

The yaw-angle control does not influence the translational dynamics of the quadrotor and, thus, can be controlled independently. First, we compute the current yaw angle ψ from the quadrotor orientation $\mathbf{R}$. Second, we compute the desired angular rate in world z direction with a proportional controller on the yaw angle error

$$r_{World} = p_{yaw} \left(\psi_{ref} - \psi \right). \tag{A.14}$$

The resulting desired rate can then be converted into the quadrotor body frame using its current orientation

$$r_{des} = \mathbf{R}_{3,3} \cdot r_{World}. \tag{A.15}$$

Low-Level Control

The commands sent to the low-level control on the PX4 are the desired body rates $\boldsymbol{\omega}_{des}$ and the desired mass-normalized thrust c_{des}. From these, the desired rotor thrusts are then computed using a feedback linearizing control scheme with the closed-loop dynamics of a first-oder system. First, the desired torques $\boldsymbol{\tau}_{des}$ are computed as

$$\boldsymbol{\tau}_{des} = \mathbf{J} \begin{bmatrix} p_{pq}(p_{des} - p) \\ p_{pq}(q_{des} - q) \\ p_r(r_{des} - r) \end{bmatrix} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}. \tag{A.16}$$

Then, we can plug $\boldsymbol{\tau}_{des}$ and c_{des} into Equations (A.7) and (A.6) and solve them for the desired rotor thrusts which have to be applied.

A.5.4 Calibration

Sensor Bias Calibration

For the state estimation and the control of our quadrotors, we make use of their gyros and accelerometers. Both these sensor units are prone to have an output bias that varies over time and that we, therefore, have to estimate. We noticed that the changes of these biases during one flight are negligible. This allows us to estimate them once at the beginning of a mission and then keep them constant. Thus, we do not have to estimate them on-line and can therefore reduce the size of the state in the state estimation. In the following, we will present a procedure that allows to estimate the biases during autonomous hover. Note, that the quadrotor can hover autonomously even with sensor biases. However, removing the biases increases the state estimation and tracking performance. For the sensor bias calibration, we look at the gyros and the accelerometers separately.

The gyro measurement equation reads as

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega} + \mathbf{b}_\omega + \mathbf{n}_\omega, \tag{A.17}$$

where $\tilde{\boldsymbol{\omega}}$ denotes the measured angular velocities, $\boldsymbol{\omega}$ the real angular velocities, $\mathbf{b}_\omega$ the bias, and $\mathbf{n}_\omega$ the noise of the gyros. This, in hover conditions, becomes

$$\tilde{\boldsymbol{\omega}} = \mathbf{b}_\omega + \mathbf{n}_\omega. \tag{A.18}$$

We assume the noise $\mathbf{n}_\omega$ to have zero mean and can therefore average the gyro measurements over N samples to estimate the gyro bias $\mathbf{b}_\omega$ as

$$\hat{\mathbf{b}}_\omega = \frac{1}{N} \sum_{k=1}^{N} \tilde{\boldsymbol{\omega}}_k. \tag{A.19}$$

The accelerometer measurement equation reads as

$$\tilde{\mathbf{a}} = \mathbf{c} + \mathbf{a}_{dist} + \mathbf{b}_{acc} + \mathbf{n}_{acc}, \tag{A.20}$$

where $\tilde{\mathbf{a}}$ denotes the measured accelerations, $\mathbf{c}$ the mass normalized thrust, $\mathbf{a}_{dist}$ the accelerations due to external disturbances, $\mathbf{b}_{acc}$ the bias, and $\mathbf{n}_{acc}$ the noise of the accelerometer. In hover conditions, $\mathbf{c} = -\mathbf{g}$ and we assume to have only small and zero mean disturbances, so the equation simplifies

$$\tilde{\mathbf{a}} = -\mathbf{g} + \mathbf{b}_{acc} + \mathbf{n}_{acc}. \tag{A.21}$$

As for the gyro bias, we assume the noise $\mathbf{n}_{acc}$ to have zero mean and can therefore average the accelerometer measurements over N samples to estimate the accelerometer bias $\mathbf{b}_{acc}$ as

$$\hat{\mathbf{b}}_{acc} = \frac{1}{N} \sum_{k=1}^{N} \tilde{\mathbf{a}}_k + \mathbf{g}. \tag{A.22}$$

When performing the sensor bias calibration, we sample the IMU readings over a 5 s period which is enough to provide an accurate estimate of their biases.

Thrust Calibration

When flying our quadrotors under very different conditions indoors and outdoors, we noticed that the produced rotor thrust can vary substantially. This can significantly reduce the control authority and hence the flight performance in situations where the produced thrust is low. To overcome this limitation, we estimate the actually produced thrust in flight.

The thrust f of a single rotor can be computed as

$$f = \frac{1}{2} \cdot \rho \cdot \Omega^2 \cdot C \cdot A, \tag{A.23}$$

where ρ is the air density, Ω is the rotor speed, C is the lift coefficient, and A is the rotor area. The rotor speed Ω is the input parameter, through which we control the

thrust. We assume that the rotor speed is controlled by the motor controllers such that we can neglect differences of the battery voltage. However, the density of the air ρ is not constant as it depends on the air pressure and temperature. Additionally, wear and possible damages to the rotors might cause unexpected changes in C and A. These three values are difficult to measure but we can estimate them together in hover flight. To do so, we first combine the three parameters and write Equation (A.23) as

$$f = G\Omega^2. \tag{A.24}$$

We refer to this equation as thrust mapping, i.e. the mapping of the rotor speed to the resulting rotor thrust. Under nominal conditions, this thrust mapping can be estimated (e.g. with a load cell) to obtain the nominal coefficient $\check{G}$ leading to a nominal thrust mapping $\check{f} = \check{G}\Omega^2$. Due to the multiplicative nature of Equation (A.23) and correspondingly (A.24), we can express the real thrust mapping coefficient as

$$G = \lambda\check{G}, \tag{A.25}$$

and hence the real produced thrust as

$$f = \lambda\check{f}. \tag{A.26}$$

This formulation allows us to estimate λ in hover and therefore calibrate the thrust mapping. For the quadrotor to hover, we know that $\boldsymbol{\tau} \overset{!}{=} 0$ and $c \overset{!}{=} g$. Thus, from Equations (A.7) and (A.6) we obtain the following matrix equation

$$\begin{bmatrix} d & -d & -d & d \\ -d & -d & d & d \\ \kappa & -\kappa & \kappa & -\kappa \\ 1/m & 1/m & 1/m & 1/m \end{bmatrix} \begin{bmatrix} \check{f}_1\lambda_1 \\ \check{f}_2\lambda_2 \\ \check{f}_3\lambda_3 \\ \check{f}_4\lambda_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g \end{bmatrix}, \tag{A.27}$$

where $d = \frac{\sqrt{2}}{2}l$. This system of equations can be solved for $\lambda_{1\ldots4}$. The nominal thrusts $\check{f}_i$ are obtained by averaging the applied nominal thrusts over N samples,

$$\check{f}_i = \frac{1}{N}\sum_{k=1}^{N}\check{f}_{i,k}. \tag{A.28}$$

To perform the thrust calibration, we sample the applied thrust commands over a 5 s period which is sufficient to get a robust thrust estimation. Note that the nominal thrust mapping is stored on the vehicle and the nominally-applied rotor thrusts $\check{f}_{i,k}$ are computed on the vehicle using the actually-commanded rotor speeds and the nominal thrust mapping.

When controlling the vehicle, we first compute the actual desired rotor thrusts, as described in Section A.5.3, which we then have to convert into the corresponding nominal rotor thrusts,

$$\check{f}_{i,des} = \frac{f_{i,des}}{\lambda_i}. \tag{A.29}$$

These nominal rotor thrusts are then converted into motor commands using the nominal thrust mapping onboard the vehicle.

A.6 Real-time Dense 3D Reconstruction

The MAV streams, through WiFi, the images $\mathbf{I}_k$ and corresponding poses $\mathbf{T}_{k,w}$ computed by SVO to the ground station at a rate of 5 Hz.[6] The observations are used to compute a dense reconstruction in real-time on the GPU. Therefore, we use the REMODE ("Regularized Monocular Depth") algorithm that we proposed in [133] and that we summarize in the following. REMODE computes dense depth maps for automatically-selected reference frames. New reference frames are selected when the Euclidean distance to the previous reference frame, normalized by the average depth in the scene, exceeds a threshold. A depth map is computed by initializing a depth-filter for every pixel in a reference view r. We use the same depth-filter formulation as in the SVO algorithm, with the difference that now every pixel of the image has a depth-filter (rather than only salient features) and that the computation is performed highly parallelized on the GPU.

Finally, smoothness on the resulting depth map is enforced by minimizing a regularized energy functional. In the following, we give an overview of the depth-filter formulation and the smoothing step.

A.6.1 Depth Filter

The depth computation for a single pixel is formalized as Bayesian estimation problem. Let the rigid body transformation $\mathbf{T}_{w,r} \in SE(3)$ describe the pose of a reference frame relative to the world frame. We initialize a depth-filter at every pixel in the reference view with high uncertainty in depth and a mean set to the average scene depth of the previous reference frame. The depth filter is described by a parametric model (A.30) that is updated on the basis of every subsequent frame k.

Given a new observation $\{\mathbf{I}_k, \mathbf{T}_{k,w}\}$ we project the 95% depth-confidence interval

[6] Although in our lab, we can transmit uncompressed, full-resolution images at rates of up to 50 Hz, we observed, during public exhibitions and outdoor experiments with more than 20-meter height, that a lower bound of 5 Hz can be usually assumed.

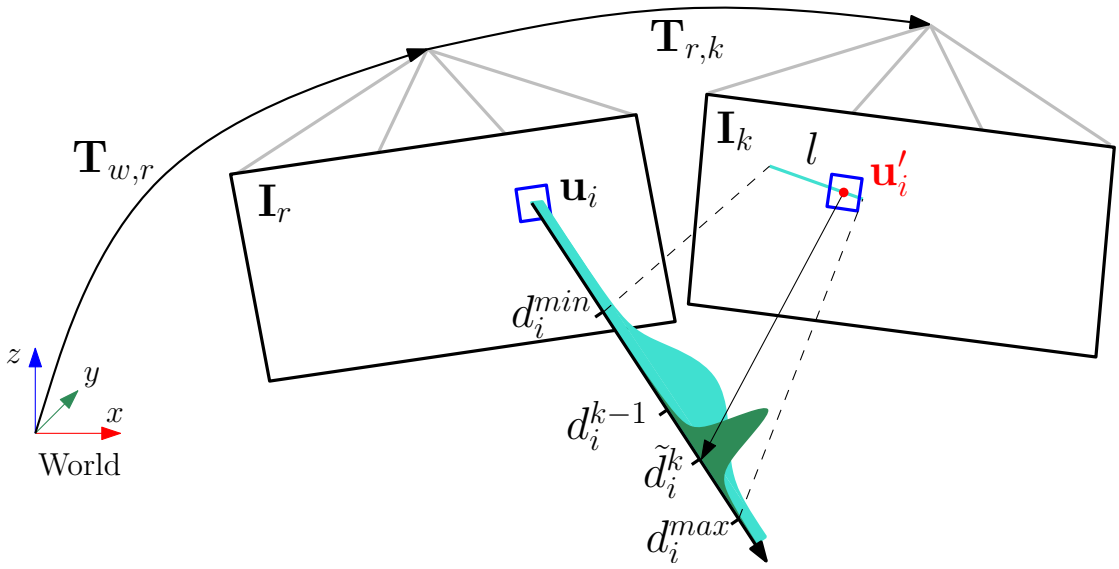


Figure A.6: Probabilistic depth estimate $\hat{d}_i$ for feature i in the reference frame r. The point at the true depth projects to similar image regions in both images (blue squares). Thus, the depth estimate is updated with the triangulated depth $\tilde{d}_i^k$ computed from the point $\mathbf{u}'_i$ of highest correlation with the reference patch. The point of highest correlation lies always on the epipolar line in the new image.

$[d_i^{\min}, d_i^{\max}]$ of the depth filter corresponding to pixel i into the image k and find a segment of the epipolar line l (see Figure A.6). Using the cross-correlation score on a 5×5 patch, we search the pixel on the epipolar line segment $\mathbf{u}'_i$ that has highest correlation with the reference pixel $\mathbf{u}_i$. A depth hypothesis $\tilde{d}_i^k$ is generated from the observation by triangulating $\mathbf{u}_i$ and $\mathbf{u}'_i$ from the views r and k respectively.

Let the sequence of $\tilde{d}_i^k$ for $k = r, \ldots, r+n$ denote a set of noisy depth measurements. As proposed in [170], we model the depth filter as a distribution that mixes a good measurement (normally distributed around the true depth d_i) and an outlier measurement (uniformly distributed in an interval $[d_i^{\min}, d_i^{\max}]$ which is known to contain the depth for the structure of interest):

$$\begin{aligned} &p(\tilde{d}_i^k | d_i, \rho_i) \\ &\quad = \rho_i \mathcal{N}(\tilde{d}_i^k | d_i, {\tau_i^k}^2) + (1-\rho_i)\mathcal{U}(\tilde{d}_i^k | d_i^{\min}, d_i^{\max}), \end{aligned} \tag{A.30}$$

where ρ_i and ${\tau_i^k}^2$ are the probability (i.e., inlier ratio) and the variance of a good measurement, respectively. Assuming independent observations, the Bayesian estimation for d_i on the basis of the measurements $\tilde{d}_i^{r+1}, \ldots, \tilde{d}_i^k$ is given by the posterior

$$p(d_i, \rho_i | \tilde{d}_i^{r+1}, \ldots, \tilde{d}_i^k) \propto p(d_i, \rho_i) \prod_k p(\tilde{d}_i^k | d_i, \rho_i), \tag{A.31}$$

with $p(d_i, \rho_i)$ being a prior on the true depth and the ratio of good measurements supporting it. A sequential update is implemented by using the estimation at time step $k-1$ as a prior to combine with the observation at time step k. To this purpose, the authors of [170] show that the posterior in (A.31) can be approximated by the product of a Gaussian distribution for the depth and a Beta distribution for the inlier ratio:

$$\begin{aligned} &q(d_i, \rho_i | a_i^k, b_i^k, \mu_i^k, {\sigma_i^k}^2) \\ &\qquad = Beta(\rho_i | a_i^k, b_i^k) \mathcal{N}(d_i | \mu_i^k, {\sigma_i^k}^2), \end{aligned} \tag{A.32}$$

where a_i^k and b_i^k are the parameters controlling the Beta distribution. The choice is motivated by the fact that the $Beta \times Gaussian$ is the approximating distribution minimizing the Kullback-Leibler divergence from the true posterior (A.31). We refer to [170] for an in depth discussion and formalization of this Bayesian update step.

A.6.2 Depth Smoothing

In [133], we introduced a fast smoothing step that takes into account the measurement uncertainty to enforce spatial regularity and mitigates the effect of noisy camera localization.

We now detail our solution to the problem of smoothing the depth map $D(\mathbf{u})$. For every pixel $\mathbf{u}_i$ in the reference image $\mathbf{I}_r : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$ the depth estimation and its confidence upon the k-th observation are given, respectively, by μ_i^k and σ_i^k in (A.32). We formulate the problem of computing a de-noised depth map $F(\mathbf{u})$ as the following minimization:

$$\min_F \int_\Omega \{ G(\mathbf{u}) \left\| \nabla F(\mathbf{u}) \right\|_\epsilon + \lambda \left\| F(\mathbf{u}) - D(\mathbf{u}) \right\|_1 \} \, d\mathbf{u}, \tag{A.33}$$

where λ is a free parameter controlling the trade-off between the data term and the regularizer, and $G(\mathbf{u})$ is a weighting function related to the "G-Weighted Total Variation", introduced in [21] in the context of image segmentation. We penalize non-smooth surfaces by making use of a regularization term based on the Huber norm of the gradient, defined as:

$$\left\| \nabla F(\mathbf{u}) \right\|_\epsilon = \begin{cases} \frac{||\nabla F(\mathbf{u})||_2^2}{2\epsilon} & \text{if } ||\nabla F(\mathbf{u})||_2 \leq \epsilon, \\ ||\nabla F(\mathbf{u})||_1 - \frac{\epsilon}{2} & \text{otherwise .} \end{cases} \tag{A.34}$$

We chose the Huber norm because it allows smooth reconstruction while preserving discontinuities at strong depth gradient locations [125]. The weighting function $G(\mathbf{u})$ influences the strength of the regularization and we propose to compute it on the basis

of the measure confidence for $\mathbf{u}$:

$$G(\mathbf{u}) = \mathbb{E}_\rho[q](\mathbf{u})\frac{\sigma^2(\mathbf{u})}{\sigma^2_{max}} + \{1 - \mathbb{E}_\rho[q](\mathbf{u})\}, \tag{A.35}$$

where we have extended the notation for the expected value of the inlier ratio $\mathbb{E}_\rho[q]$ and the variance σ^2 in (A.32) to account for the specific pixel $\mathbf{u}$. The weighting function (A.35) affects the strength of the regularization term: for pixels with a high expected value for the inlier ratio ρ the weight is controlled by the measurement variance σ^2; measurements characterized by a small variance (i.e. reliable measurements) will be less affected by the regularization; differently, the contribution of the regularization term will be heavier for measurements characterized by a small expected value for the inlier ratio or higher measurement variance.

The solution to the minimization problem (A.33) is computed iteratively based on the work in [27]. The algorithm exploits the primal dual formulation of (A.33) and is detailed in [133].

A.7 Results

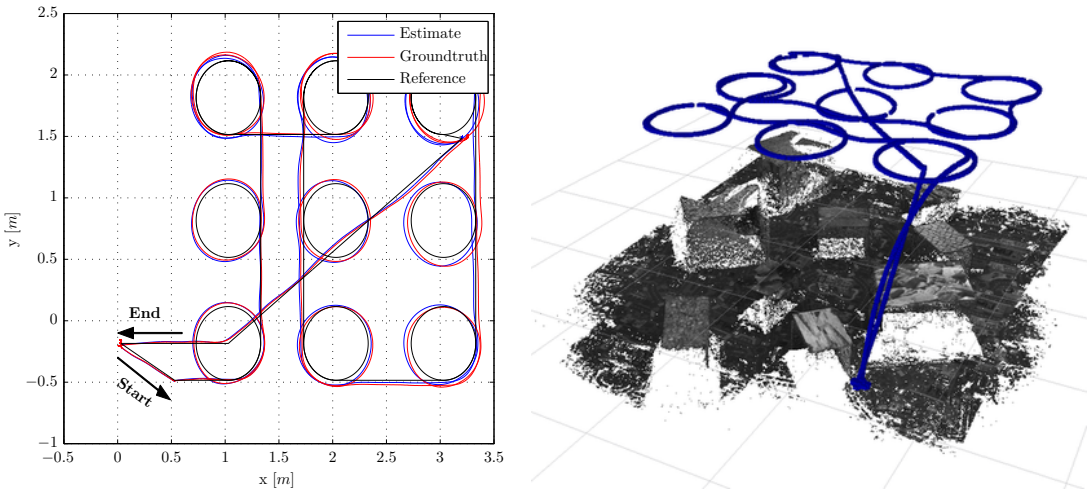
A.7.1 Indoor flight

Our flying arena is equipped with an OptiTrack motion-capture system by Natural-Point,[7] which we only used for ground-truth comparison. Its floor is covered with a texture-rich carpet and boxes for 3D structure as shown in Figure A.7. The quadrotor was requested to autonomously execute a closed-loop trajectory specified by waypoints. The trajectory was 20 meters long and the MAV flew on average at 1.7 meters above ground. Figures A.8, A.9, and A.10, show the accurate trajectory following as well as the position and orientation estimation errors, respectively. For ground-truth comparison, we aligned the first 2 meters of the estimated trajectory to the ground truth [169]. The maximum recorded position drift was 0.5% of the travelled distance. As observed, the quadrotor was able to return very close to the start point (with a final absolute error smaller than 5 cm). This result was confirmed in more than 100 experiments run at public events, exhibitions, and fairs. These results outperform those achieved with MAVs based on PTAM. This is due to the higher precision of SVO. Comparisons between SVO and PTAM are reported in [49].

Apart from its higher precision and frame rate, another main advantage of our SVO compared to PTAM is its robustness in scenes with repetitive, and high-frequency textures (e.g., asphalt, grass), c.f. Figure A.11. Figure A.12 shows a comparison of the map generated with PTAM and SVO in the same scene. While PTAM generates outlier

[7] http://www.naturalpoint.com/optitrack/

Figure A.7: Our flying arena equipped with an OptiTrack motion-capture system (for ground-truth recording), carpets and boxes for 3D structure.



(a) (b)

Figure A.8: Comparison of estimated and ground-truth position to a reference trajectory (a) flown indoors over a scene as reconstructed in (b). A photograph of a similar scene is shown in Figure A.7.

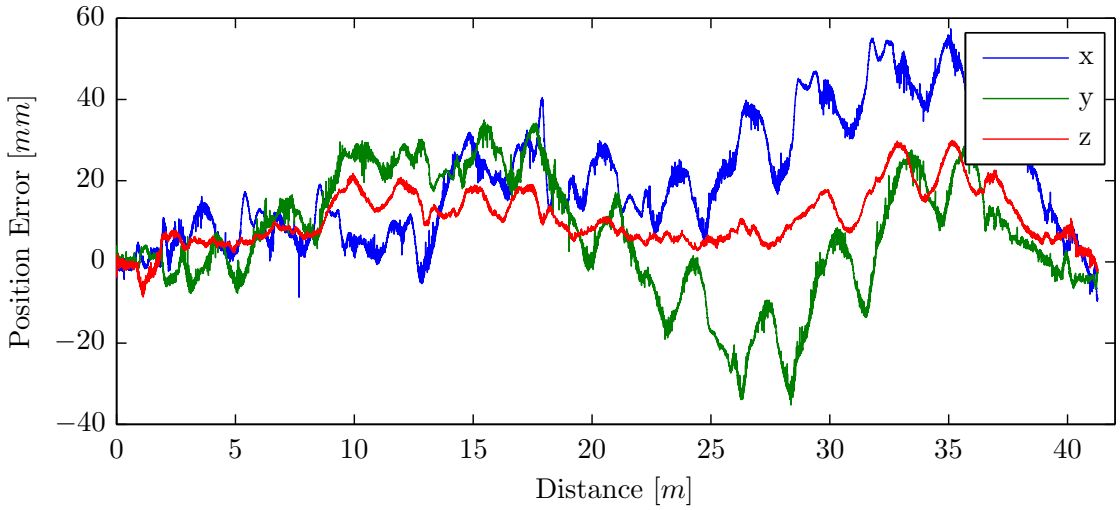


Figure A.9: Error between estimated and ground-truth position for the trajectory illustrated in Figure A.8.

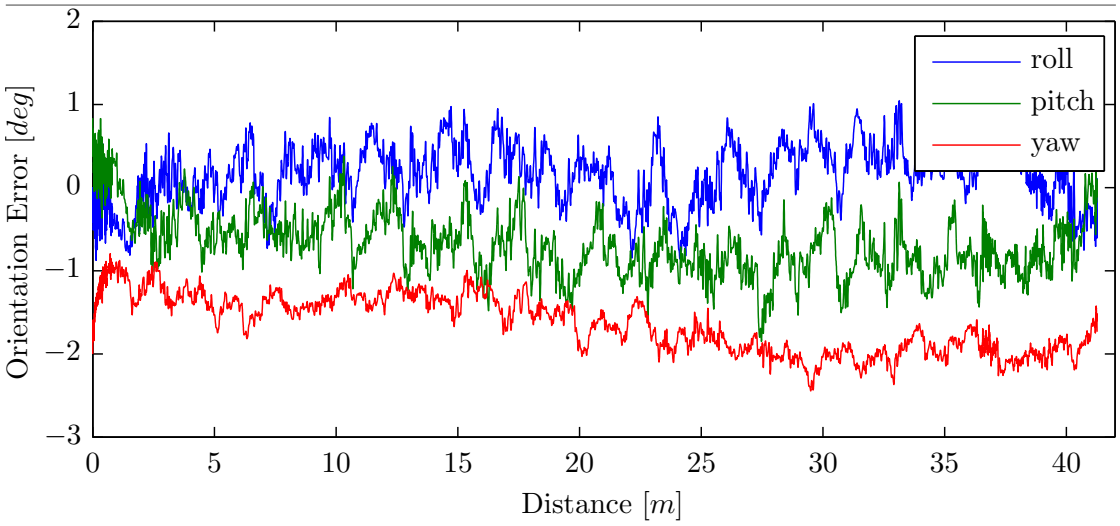


Figure A.10: Error between estimated and ground-truth orientation for the trajectory illustrated in Figure A.8.

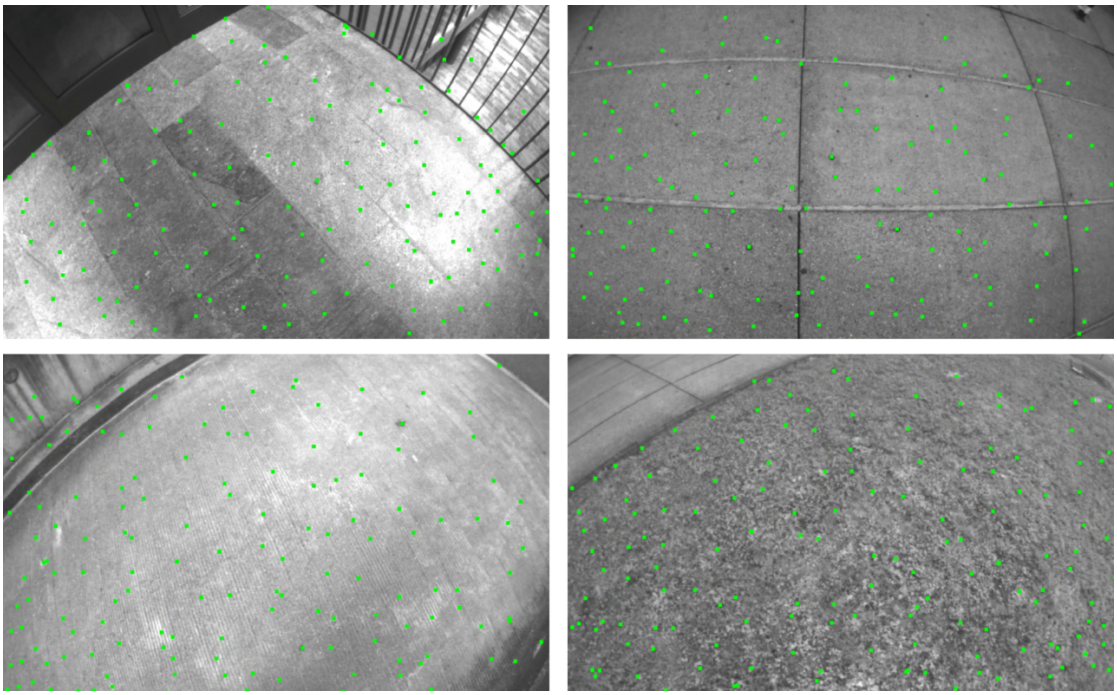


Figure A.11: Successful tracking in scenes of high-frequency texture.

3D points, by contrast SVO has almost no outliers thanks to the use of the depth-filter.

A.7.2 Outdoor flight

An outdoor demonstration was performed at the Zurich firefighter training area in front of real firemen. This area features a large mock-up disaster site, consisting of two main towers and several stone blocks gathered on the ground. Due to the unavailability of a reliable GPS signal in this area, results are not compared to GPS. The quadrotor was requested to reach a height of 20 m and follow a piece-wise straight trajectory (c.f., Figure A.13). The overall trajectory length was 140 m. The first 100 meters of this trajectory were executed fully autonomously and are indicated in blue. After the autonomous flight, we handed a joypad game controller to a fireman with no pilot experience. The joypad was connected to the ground station and allowed sending the quadrotor simple up-down, left-right, forward-backward velocity commands expressed in the world reference frame. Thanks to the assistance of the vision-based control running onboard the quadrotor, the firefighter was able to successfully and easily control the vehicle back to his position.

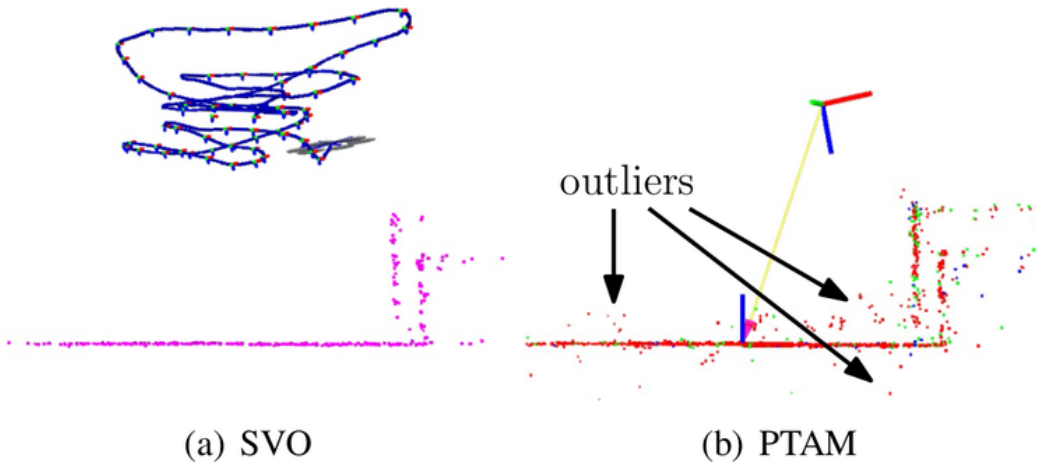


Figure A.12: Side-view of a piecewise-planar map created by SVO and PTAM. The proposed method has fewer outliers due to the depth-filter.

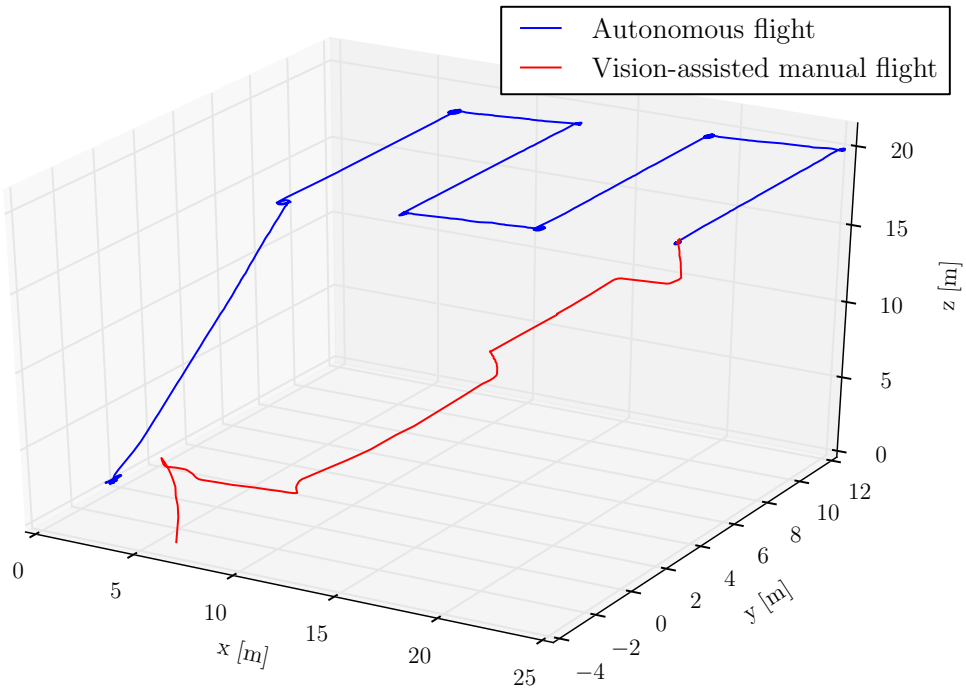
A.7.3 Reconstruction

The laptop that served as a groundstation to run the live, dense 3D reconstruction is a Lenovo W520 with an Intel i7-3720QM processor, equipped with 16 GB of RAM, and an NVIDIA Quadro K2000M GPU with 384 CUDA cores.

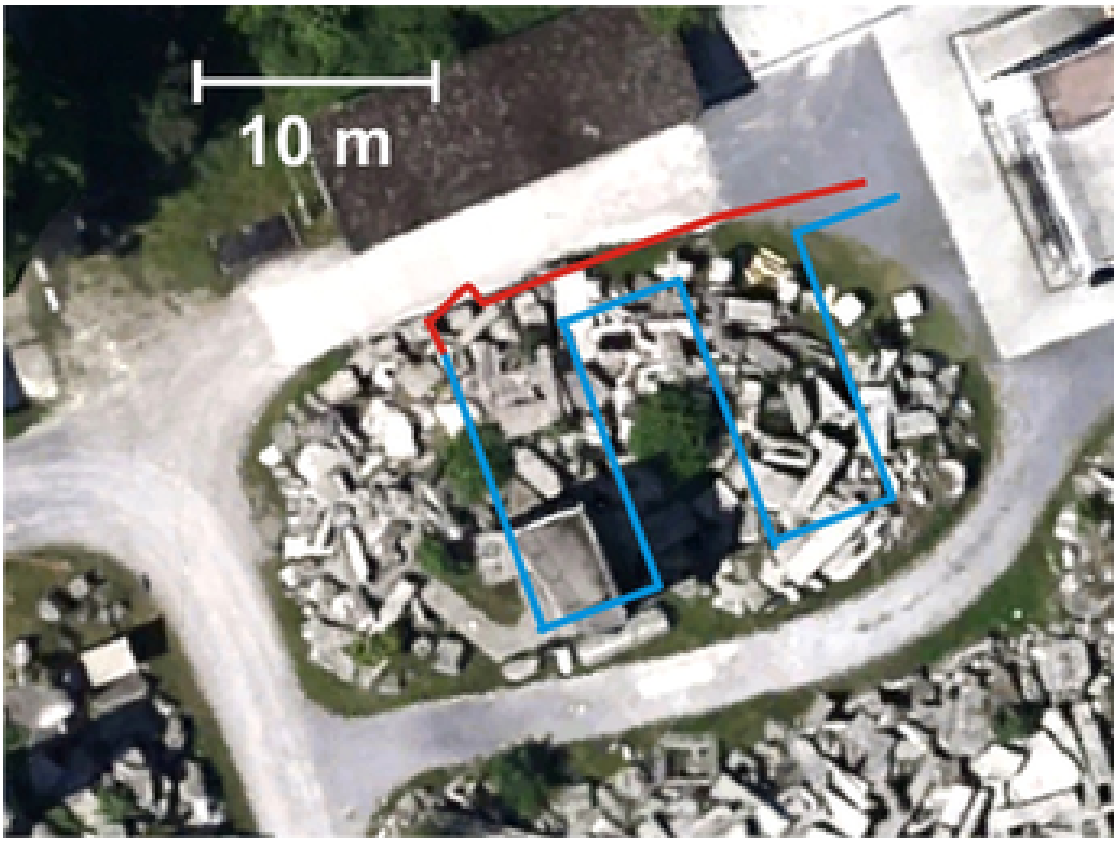
Figures A.14 and A.15 show the dense reconstruction results from an indoor and outdoor scene, respectively.

To quantitatively evaluate our approach, we tested REMODE in [133] on a synthetic dataset provided in [62]. Figure A.16 reports the main results of the reconstruction performance. The dataset consisted of views generated through ray-tracing from a three-dimensional synthetic model. The evaluation was based on a comparison with the ground truth depth map corresponding to the view taken as reference in the reconstruction process. As an evaluation metrics, we used the percentage of ground truth depths that have been estimated by the proposed method within a certain error (see Figure A.17). To show the effectiveness of our approach, we compared our result with the depth map computed according to the state-of-the-art method introduced in [170].

Our approach was capable to recover a number of erroneous depth estimations, thus yielding a sensible improvement in terms of completeness. To verify the robustness against noisy camera-pose estimation, we corrupted the camera position with Gaussian noise, with zero mean and one-centimeter standard deviation on each coordinate. The results show that the completeness drops. This is inevitable due to the smaller number of converged estimations. However, the computation of the depth map takes advantage of the de-noising step.

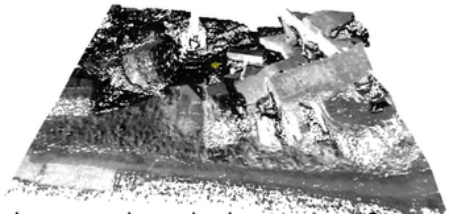


(a) Side view.



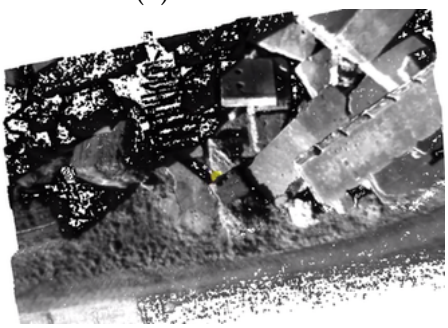
(b) Top view.

Figure A.13: Outdoor trajectory of 140 meters. Blue denotes the trajectory executed autonomously, red the one executed manually by a firefighter with no pilot experience assisted by the onboard vision-based controller.



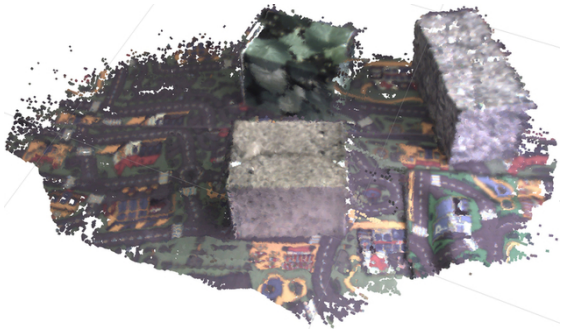
(a) Side view.

(b) Side view.

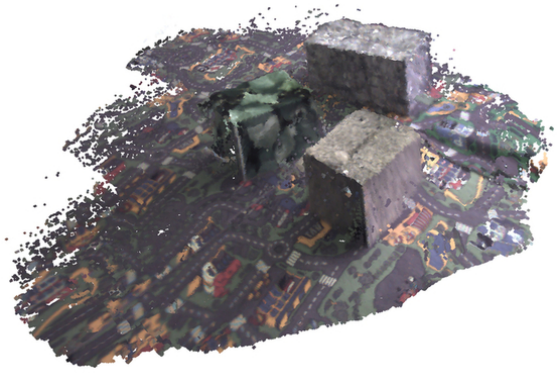


(c) Top view.

Figure A.14: Outdoor, dense 3D reconstruction.



(a) Side view.



(b) Side view.

Figure A.15: Indoor, dense 3D reconstruction

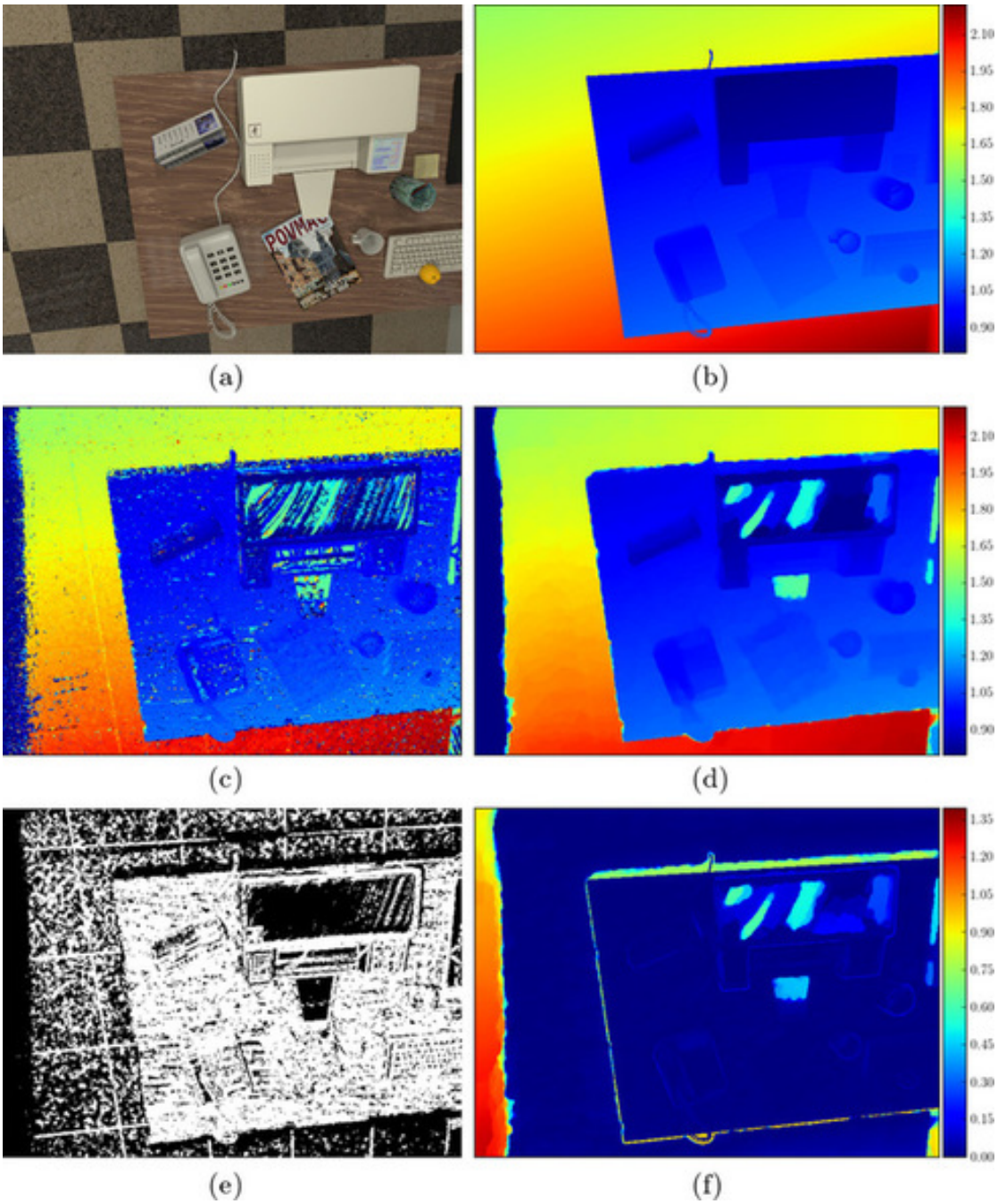


Figure A.16: Evaluation of dense reconstruction on synthetic dataset. (a): the reference view. (b): ground truth depth map. (c): depth map based on [170]. (d): depth map computed by REMODE. (e): map of reliable measurements (white pixels are classified as reliable). (f): error of REMODE.

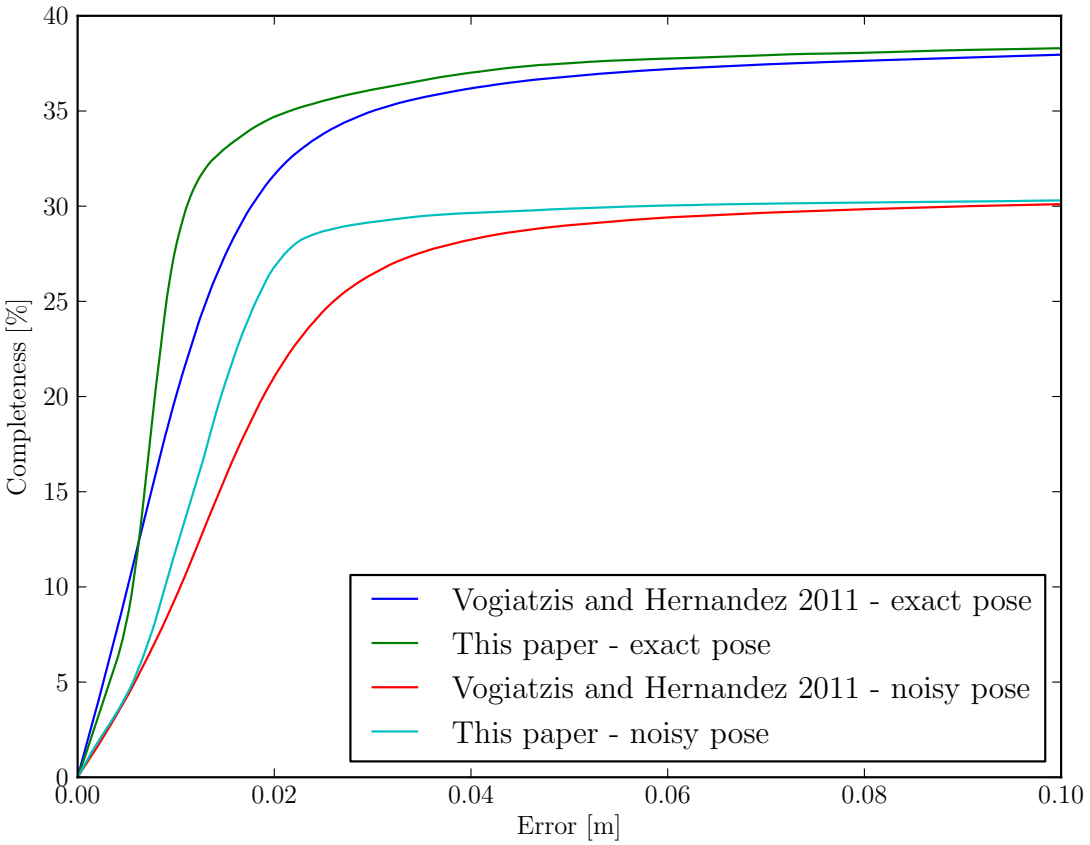


Figure A.17: The completeness of the synthetic experiment, i.e., the percentage of ground truth measurements that are within a certain error from the converged estimations. For color blind readers: top line, this paper - exact pose; second line from the top, Vogiatzis and Hernandez 2011 - exact pose; third line from the top, this paper - noisy pose; bottom line, Vogiatzis and Hernandez 2011 - noisy pose.

A.8 Discussions and Conclusion

A.8.1 Lessons Learned

System Design

The use of open-source software and hardware components was crucial to adapt them exactly to our needs. This allowed us to tune all the components such that they could work together properly and, hence, achieve a very good overall system performance. Furthermore, since we mostly used off-the-shelf hardware components, our quadrotor is relatively cheap, easy to upgrade, and fast to repair.

Indoor Experiments

In the proposed setup, the MAV relies on our visual odometry pipeline (i.e., SVO). However, several factors may disturb visual-odometry pipelines. Examples include flashlights of photographers, sudden illumination changes (e.g., when moving from a shadow area to an illuminated area), too rapid motion of the vehicle, or poor texture on the ground. If the visual odometry cannot recover within a certain amount of time, it is not possible to continue the mission and the quadrotor has to land. Since this can happen often in an experimental phase, it is crucial to have a safe landing procedure such that the quadrotor does not get damaged. To do so, we implemented an open-loop emergency-landing procedure, where the quadrotor stabilizes its attitude with the IMU and applies a thrust to slowly descend to the ground. An open-loop procedure is necessary since, especially in indoor environments, fall-backs, such as GPS, are not available.

Outdoor Experiments

In outdoor experiments, we noticed that the produced thrust can vary significantly due to different air temperature and pressure. This can have noticeable effects on the flight performance. Especially for an open-loop landing maneuver, after losing visual tracking it is crucial to know what the actually-produced thrust is. For this reason, we implemented a routine to estimate the produced thrust (see Section A.5.4) in the beginning of a mission during a short period of hover flight. With this routine we can account for different air densities and possible damage of the rotors.

In addition, outdoor experiments on partially-cloudy days showed that it is crucial to have a camera with a high dynamic range. We set the camera settings before every mission and kept them fixed during the mission due to the weak performance of the camera-driver's auto-adjustment that either flickers, overexposes, or underexposes the scene. This can cause the visual odometry to lose tracking when the quadrotor is

transitioning between dark and bright scenes.

Dense Reconstruction

In the current reconstruction pipeline, the depthmaps computed by REMODE are visualized as a point-cloud at the pose computed by SVO. However, as SVO drifts over time, the depth-maps are not perfectly aligned. Therefore, in future work we will integrate REMODE in our collaborative mapping optimisation back-end [45] in order to compute globally-optimized maps. Additionally, in order to minimize the noise, we will apply a depthmap-fusion stage such as in [46]. Finally, depending on the height, the structure and the appearance of the scene, different motions are required to perform the depthmap reconstruction as fast as possible. Therefore, in [48], we proposed an active mapping formalisation that we will integrate tightly into the current system.

A.8.2 Conclusion

We presented a system for mapping an unknown indoor or outdoor environment from the air in real time. Our system consists of an autonomous vision-based quadrotor and a ground station laptop for dense 3D reconstruction. The quadrotor can fly fully autonomously without relying on any external positioning system, which allows it to fly in unknown indoor and outdoor environments. This is crucial for search-and-rescue where one cannot rely on any functional infrastructure. Furthermore, the autonomy of our quadrotor allows non-expert operators to use our system with no training, as we demonstrated in the outdoor firefighter training area. From the images streamed by the quadrotor, we compute a live, dense 3D map on the ground station. Since the 3D map computation is performed in real time, the operator gets an immediate feedback without any unnecessary delays in the mission.

In numerous (more than 100) demonstrations, as well as indoor and outdoor experiments, we showed that our system works robustly, is easy to set up, and can easily be controlled by only one operator.

B Automatic Re-Initialization and Failure Recovery

M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza. "Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 1722–1729. DOI: 10.1109/ICRA.2015.7139420

Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor

Matthias Faessler, Flavio Fontana, Christian Forster and Davide Scaramuzza

Abstract — Autonomous, vision-based quadrotor flight is widely regarded as a challenging perception and control problem since the accuracy of a flight maneuver is strongly influenced by the quality of the on-board state estimate. In addition, any vision-based state estimator can fail due to the lack of visual information in the scene or due to the loss of feature tracking after an aggressive maneuver. When this happens, the robot should automatically re-initialize the state estimate to maintain its autonomy and, thus, guarantee the safety for itself and the environment. In this paper, we present a system that enables a monocular-vision–based quadrotor to automatically recover from any unknown, initial attitude with significant velocity, such as after loss of visual tracking due to an aggressive maneuver. The recovery procedure consists of multiple stages, in which the quadrotor, first, stabilizes its attitude and altitude, then, re-initializes its visual state-estimation pipeline before stabilizing fully autonomously. To experimentally demonstrate the performance of our system, we aggressively throw the quadrotor in the air by hand and have it recover and stabilize all by itself. We chose this example as it simulates conditions similar to failure recovery during aggressive flight. Our system was able to recover successfully in several hundred throws in both indoor and outdoor environments.

B.1 Introduction

B.1.1 Motivation

Autonomous Micro Aerial Vehicles (MAVs) will soon play a major role in remote inspection and search-and-rescue missions. In these applications, the MAVs will have to operate in unknown indoor and outdoor environments, which prevents them from relying on external positioning systems (e.g. GPS or motion-capture systems). A viable solution to maintain position tracking for lightweight MAVs is to use on-board cameras. Unfortunately, vision algorithms are prone to lose visual tracking during fast motions, e.g., when executing aggressive maneuvers, or under strong illumination changes that can occur when transitioning from dark to bright scenes. When visual tracking is lost, the vehicle typically has to descend and land in a partially open-loop maneuver, or a trained pilot has to take over control. To re-initialize the vision pipeline, manual procedures (by hand or remote control) are required by the operators, which renders re-initialization during flight very difficult or even impossible.

In this paper, we describe an approach to allow a monocular-vision–based quadrotor to recover completely autonomously from difficult conditions, where it has lost visual tracking, and automatically re-initialize its vision pipeline during flight. This enables the quadrotor to recover in case of a failure of the vision pipeline and continue its mission without landing. Some snapshots of an autonomous recovery outdoors are shown in Fig. B.1. When performing aggressive maneuvers, our system allows us to push the quadrotor to its limits and beyond, while still being able to recover at any time without resorting to any external pose-estimation fall-back. Along with re-initializing in flight, our system enables instant launches of the quadrotor by manually throwing it in the air. This starting procedure is not only very quick, but also enables an untrained operator to start the quadrotor without remote control.

B.1.2 Related Work

In the recent years, several groups have demonstrated MAVs that can perform impressive aerobatics [1, 97], pass through narrow gaps [107], or recover and stabilize virtually from any initial condition [1, 107]. However, besides being limited to a set of learned maneuvers, the platforms used in these demonstrations relied on the accurate and high-frequency position estimates provided by external cameras (such as Vicon[1] or custom-made motion-capture systems) and off-board computation. Nonetheless, these systems need pre-installation and calibration of the cameras and, therefore, cannot be used in unknown and yet-unexplored environments.

To the best of our knowledge, aggressive maneuvers similar to the works mentioned

[1] www.vicon.com

(a) $t = 0\,\text{ms}$ **(b)** $t = 80\,\text{ms}$ **(c)** $t = 440\,\text{ms}$

(d) $t = 1{,}120\,\text{ms}$ **(e)** $t = 1{,}640\,\text{ms}$ **(f)** $t = 2{,}000\,\text{ms}$

Figure B.1: Autonomous recovery after throwing the quadrotor by hand: (a) the quadrotor detects free fall and (b) starts to control its attitude to be horizontal. Once it is horizontal, (c) it first controls its vertical velocity and then, (d) its vertical position. The quadrotor uses its horizontal motion to initialize its visual-inertial state estimation and uses it (e) to first break its horizontal velocity and then (f) lock to the current position.

above have not yet been achieved with autonomous quadrotors that rely on on-board sensing and on-board computation, since their state estimate is not as precise and reliable as the one from external positioning systems. To overcome the limitation of being restricted to the volume of a motion-capture system, on-board sensors, such as cameras and Inertial Measurement Units (IMU), are the only viable solution for lightweight MAVs, as demonstrated in [172, 151, 145]. However, current vision-based quadrotors still operate in near-hover conditions, with only few attempts, such as [155], to perform more aggressive maneuvers.

If a quadrotor's vision pipeline fails, there is typically a small set of options left: (i) a pilot must take over; (ii) the quadrotor must land immediately; (iii) the quadrotor must use simple fall backs for stabilization (e.g., based on optical flow algorithms [172]), which do not allow the continuation of its mission without further actions. In [151], a linear sliding window formulation for monocular visual-inertial systems was presented to make a vision-based quadrotor capable of failure recovery and on-the-fly initialization. However, this work assumed that visual features could be extracted and correctly tracked right from the beginning of the recovery procedure.

Along with possible failures of their state-estimation pipeline, monocular-vision–based quadrotors present the drawback that they typically require an initialization phase before they can fly autonomously. This initialization is usually performed by moving the quadrotor by hand or through remote control. Since this is time consuming and not easy to perform, attempts have been made to perform the initialization automatically. For instance, in [23], the authors presented a system that allows the user to toss a quadrotor in the air, where it then initializes a visual-odometry pipeline. Nevertheless, that system still required several seconds for the state estimate to converge before the toss and several more seconds until the visual-odometry pipeline was initialized. A closed-form solution for state estimation with a visual-inertial system that does not require initialization was presented in [103]. However, this approach is not suitable for systems that rely on noisy sensor data.

B.1.3 Contributions and Outline

We present a system that enables a monocular vision-based quadrotor to autonomously recover from state-estimation failures quickly, and re-initialize its visual-inertial state estimation. The described system allows the quadrotor to recover from any attitude, even with high linear velocities and body rates. The performance of our recovery strategy is evaluated in the scenario where a quadrotor is thrown in the air by hand and must stabilize based only on its on-board sensors. We present an attitude estimator based on quaternions which fuses measurements from the gyroscopes and accelerometers to obtain a globally-valid attitude estimate at the time when it is launched. This allows the user to throw the quadrotor with any initial attitude, and the controller immediately starts to guide it to horizontal position. In contrast to [151], our system does not require the observation of visual features at the beginning of the recovery procedure but only once its attitude is stabilized, which simplifies feature tracking greatly and reduces computational complexity. In addition to [23], no preparation time before launching the quadrotor is required and the entire recovery is performed more quickly.

Along with a very quick and easy start by simply throwing the quadrotor in the air, our system enables more aggressive flight, where a vision-based quadrotor can perform a maneuver with the expectation that it will lose tracking but still regain control.

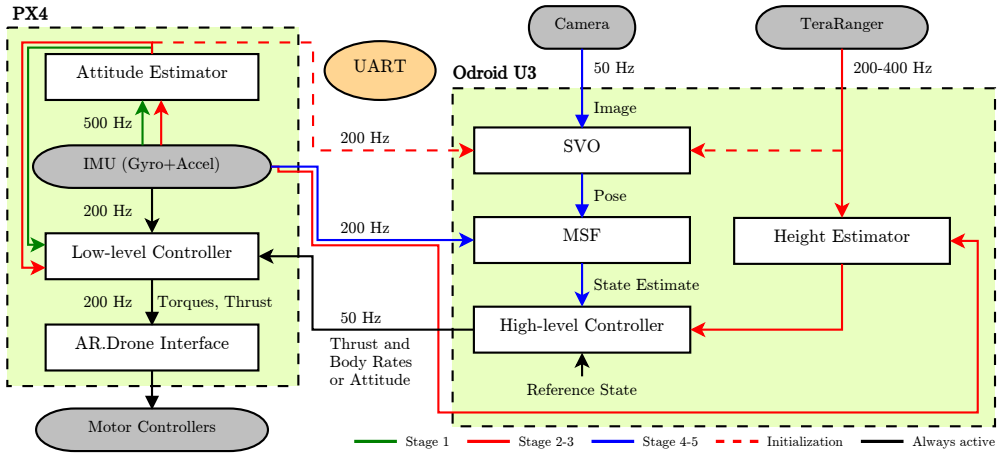


Figure B.2: Control-system overview: the PX4 and Odroid U3 communicate with each other over a UART interface. Gray boxes are sensors and actuators; white boxes depict software modules. Green arrows indicate the communication that is required specifically during stage 1, red arrows indicate the communication that is required specifically during stages 2 and 3, and blue arrows indicate the communication that is required during stages 4 and 5 as well as for normal vision-based flight. The dashed red arrows indicate measurements that are used only once for initializing SVO. Black arrows indicate communication that is required in all the recovery stages as well as for normal vision-based flight. A more detailed description of the hardware set-up is given in Section B.4.1

The remainder of this paper is organized as follows. Section B.2 describes the implemented high-level and low-level controllers, as well as estimation algorithms used for recovery. Section B.3 describes the different stages of our recovery procedure. Section B.4 introduces our quadrotor platform and presents the experimental results. Finally, Section B.5 concludes the work.

B.2 Control and State Estimation

This section describes our control and state estimation algorithms that are used for recovery as well as for vision-based flying with our quadrotor. The controller is split into a high-level part and a low-level part. The high-level controller enables the quadrotor to track desired positions and velocities, whereas the low-level controller enables it to track desired attitudes or body rates. The overall state estimation and control structure with the used sensors is illustrated in Fig. B.2.

B.2.1 Nomenclature

When describing the control and state estimation, we make use of some notation that we introduce here for clarity. We use a hat (e.g. $\hat{v}$) and a tilde (e.g. $\tilde{v}$) to denote an

estimated and measured value, respectively. To describe the multiplication of two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ we write $\mathbf{q}_1 \otimes \mathbf{q}_2$, and we write $\mathbf{q} \odot \mathbf{v}$ for the rotation of a vector $\mathbf{v}$ by the quaternion $\mathbf{q}$. Furthermore, we express the basis vectors of a coordinate system as e.g. $\mathbf{e}_x^B$ which denotes the x-basis vector of the body coordinate system, as illustrated in Fig. B.3. A prescript (e.g. ${}_B\mathbf{v}$) indicates that the vector $\mathbf{v}$ is expressed in the body coordinate system. Vectors without prescripts are expressed in the world coordinate system with the exception of the body rates $\boldsymbol{\omega}$, which are always expressed in body coordinates.

B.2.2 Dynamical Model

For state estimation and control, we make use of the following dynamical model for our quadrotor:

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{B.1}$$

$$\dot{\mathbf{v}} = \mathbf{g} + \mathbf{q} \odot \mathbf{c}, \tag{B.2}$$

$$\dot{\mathbf{q}} = \boldsymbol{\Lambda}(\boldsymbol{\omega}) \cdot \mathbf{q}, \tag{B.3}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} \cdot (\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}), \tag{B.4}$$

where $\mathbf{r} = [x \ y \ z]^T$ and $\mathbf{v} = [v_x \ v_y \ v_z]^T$ are the position and velocity in world coordinates, $\mathbf{q} = [q_w \ q_x \ q_y \ q_z]^T$ is the orientation of the quadrotor's body coordinates with respect to the world coordinates, and $\boldsymbol{\omega} = [p \ q \ r]^T$ denotes the body rates (roll, pitch and yaw, respectively) expressed in body coordinates. The skew-symmetric matrix $\boldsymbol{\Lambda}(\boldsymbol{\omega})$ is defined as

$$\boldsymbol{\Lambda}(\boldsymbol{\omega}) = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}. \tag{B.5}$$

We define the gravity vector as $\mathbf{g} = [0 \ 0 \ -g]^T$ with $g = 9.81\,\mathrm{m\,s^{-2}}$ and the second-order moment-of-inertia matrix of the quadrotor as $\mathbf{J} = \mathrm{diag}\left(J_{xx}, J_{yy}, J_{zz}\right)$. The mass-normalized thrust vector is $\mathbf{c} = [0 \ 0 \ c]^T$, with

$$mc = f_1 + f_2 + f_3 + f_4, \tag{B.6}$$

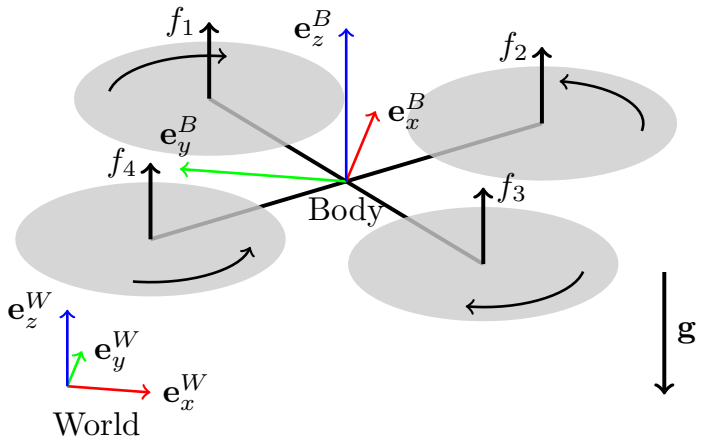


Figure B.3: Quadrotor with coordinate system and rotor forces.

where m is the mass of the quadrotor and f_i are the four motor thrusts as illustrated in Fig. B.3. The torque inputs $\boldsymbol{\tau}$ are composed of the single-rotor thrusts as

$$\boldsymbol{\tau} = \begin{bmatrix} \frac{\sqrt{2}}{2} l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} l(-f_1 - f_2 + f_3 + f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \tag{B.7}$$

where l is the quadrotor arm length and κ is the rotor-torque coefficient.

Our coordinate-system conventions and rotor numbering are illustrated in Fig B.3.

B.2.3 High-Level Controller

The high-level controller takes a reference state and a state estimate as input and computes the desired attitude or desired body rates, which are then sent to the low-level controller. A reference state consists of a reference position $\mathbf{r}_{ref}$, a reference velocity $\mathbf{v}_{ref}$, a reference acceleration $\mathbf{a}_{ref}$, and a reference heading ψ_{ref}. First, we describe the position controller, followed by the attitude controller.

Position Controller

To track a reference trajectory, we implemented a PD controller with feed-forward terms on the reference acceleration from the trajectory and gravity:

$$\mathbf{a}_{des} = \mathbf{P}_{pos} \cdot \left(\mathbf{r}_{ref} - \hat{\mathbf{r}}\right) + \mathbf{D}_{pos} \cdot \left(\mathbf{v}_{ref} - \hat{\mathbf{v}}\right) + \mathbf{a}_{ref} - \mathbf{g}, \tag{B.8}$$

with gain matrices $\mathbf{P}_{pos} = \text{diag}\left(p_{xy}, p_{xy}, p_z\right)$ and $\mathbf{D}_{pos} = \text{diag}\left(d_{xy}, d_{xy}, d_z\right)$. To compute the desired normalized thrust c_{des}, we project the desired acceleration onto the current ${}_W\mathbf{e}_z^B$ axis

$$c_{des} = \mathbf{a}_{des} \cdot \mathbf{e}_z^B. \tag{B.9}$$

The output of the position controller is the desired accelerations $\mathbf{a}_{des}$. The desired acceleration, together with the reference heading ψ_{ref}, encodes the desired orientation as well as a mass normalized thrust c_{des}.

Attitude Controller

Since a quadrotor can only accelerate in its body z direction, $\mathbf{a}_{des}$ enforces two degrees of freedom of the desired attitude. The third degree of freedom is enforced by the reference heading ψ_{ref}. Note that the rotation around the body z axis has no influence on the translational behaviour of the quadrotor. Therefore, we want to align the body z axis with the desired acceleration $\mathbf{a}_{des}$ by rotating around the body x and y axes and use rotations around the body z axis only to control the heading. Our quadrotors have much more attitude control authority on the x and y axes than on the z axis because there, they can make use of thrust differences as opposed to differences in rotor drag torques. The moment due to the maximum-possible thrust difference is much larger than the maximum possible difference of rotor drag torques. For this reason, we separate the attitude control into two parts as described in the following.

Desired Roll and Pitch Rates From the current attitude estimate and the desired acceleration, we can compute the current and the desired body z axes, respectively, as

$$\hat{\mathbf{e}}_z^B = \hat{\mathbf{q}} \otimes [0\ 0\ 1]^T, \quad \mathbf{e}_{z,des}^B = \frac{\mathbf{a}_{des}}{\|\mathbf{a}_{des}\|}. \tag{B.10}$$

Now, we design an error quaternion that describes the necessary rotation to align these two vectors. To do so, we compute the angle α between the two vectors and a normal vector $\mathbf{n}$ to both of them:

$$\alpha = \arccos(\hat{\mathbf{e}}_z^B \cdot \mathbf{e}_{z,des}^B), \tag{B.11}$$

$$\mathbf{n} = \frac{\hat{\mathbf{e}}_z^B \times \mathbf{e}_{z,des}^B}{\|\hat{\mathbf{e}}_z^B \times \mathbf{e}_{z,des}^B\|}. \tag{B.12}$$

Since we want to apply this rotation with respect to the current body orientation, we have to transform the rotation axis $\mathbf{n}$ into body coordinates using the current attitude estimate $\hat{\mathbf{q}}$

$${}_B\mathbf{n} = \hat{\mathbf{q}}^{-1} \odot \mathbf{n}. \tag{B.13}$$

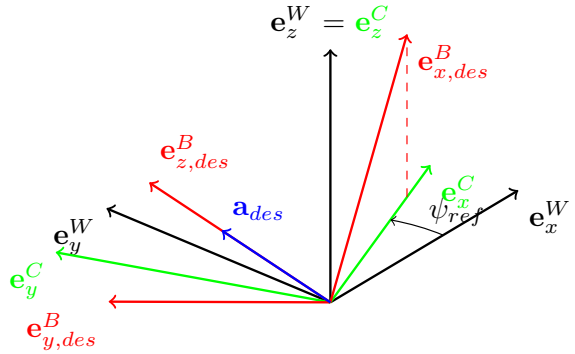


Figure B.4: Coordinate frames used in the attitude controller. We make use of the coordinate frame C, which is obtained by rotating the world frame (W) by the reference heading ψ_{ref} to construct, together with the desired acceleration $\mathbf{a}_{des}$, the full desired orientation of the body frame (B).

The error quaternion can then be constructed as

$$\mathbf{q}_{e,rp} = \begin{bmatrix} \cos(\frac{\alpha}{2}) \\ {}_B\mathbf{n}\sin(\frac{\alpha}{2}) \end{bmatrix}. \tag{B.14}$$

Note that if $\alpha = 0$, the rotation axis is undetermined and we set the error quaternion $\mathbf{q}_{e,rp}$ to be the identity directly. Also note that by construction, the z component of $\mathbf{q}_{e,rp}$ is always zero, which assures that no rotation around the body z axis is necessary to align $\hat{\mathbf{e}}_z^B$ with $\mathbf{e}_{z,des}^B$. From the error quaternion $\mathbf{q}_{e,rp}$, we can then compute the desired roll and pitch rates with the following control law:

$$\begin{bmatrix} p \\ q \end{bmatrix}_{des} = \begin{cases} 2 \cdot p_{rp} \cdot \mathbf{q}_{e,rp}^{(x,y)} & \text{if } \mathbf{q}_{e,rp}^{(w)} \geq 0 \\ -2 \cdot p_{rp} \cdot \mathbf{q}_{e,rp}^{(x,y)} & \text{if } \mathbf{q}_{e,rp}^{(w)} < 0 \end{cases}. \tag{B.15}$$

It can be shown that this control law is globally asymptotically stable and its discrete implementation is robust to measurement noise [19, 104].

Desired Yaw Rate To compute the desired yaw rate r_{des}, we look at the heading error that remains after aligning $\hat{\mathbf{e}}_z^B$ with $\mathbf{e}_{z,des}^B$ with the above control law. To do so, we first compute the full desired attitude. For this, we make use of an intermediate coordinate system C, which is the world frame rotated around its z axis by the desired heading ψ_{ref} as illustrated in Fig. B.4. The x and y axes of the coordinate system C are defined as

$$\mathbf{e}_x^C = [\cos(\psi_{ref}) \ \sin(\psi_{ref}) \ 0]^T, \tag{B.16}$$
$$\mathbf{e}_y^C = [-\sin(\psi_{ref}) \ \cos(\psi_{ref}) \ 0]^T. \tag{B.17}$$

The goal of rotating around the quadrotor's z axis is to align the projection of its x axis onto the world $x-y$ plane with $\mathbf{e}_x^C$. This forces the desired body x axis $\mathbf{e}_{x,des}^B$ to lie in a plane spanned by $\mathbf{e}_x^C$ and $\mathbf{e}_z^W$, which is fulfilled by

$$\mathbf{e}_{x,des}^B = \frac{\mathbf{e}_y^C \times \mathbf{e}_{z,des}^B}{\|\mathbf{e}_y^C \times \mathbf{e}_{z,des}^B\|}. \tag{B.18}$$

Note that if this cross product is zero, there are infinitely many rotations around the desired body z axis that achieve the desired heading. Therefore, in that case, we apply a desired yaw rate $r_{des} = 0$.

If the desired body z axis has a negative z component (i.e. $\mathbf{e}_{z,des}^B$ is pointing downwards), the projection of the computed desired body x axis into the horizontal plane will point in the opposite direction of $\mathbf{e}_x^C$, therefore we use the negation of it. From $\mathbf{e}_{x,des}^B$ and $\mathbf{e}_{z,des}^B$, we can then compute $\mathbf{e}_{y,des}^B$ as

$$\mathbf{e}_{y,des}^B = \frac{\mathbf{e}_{z,des}^B \times \mathbf{e}_{x,des}^B}{\|\mathbf{e}_{z,des}^B \times \mathbf{e}_{x,des}^B\|}. \tag{B.19}$$

Now, the full desired attitude $\mathbf{q}_{des}$ can be built from the three desired body axes $\mathbf{e}_{x,des}^B$, $\mathbf{e}_{y,des}^B$ and $\mathbf{e}_{z,des}^B$. Our definition of the heading is different than in the controller presented in [19] and has the advantage of being meaningful for any orientation of the quadrotor, which is not the case, for instance, when using a definition based on Euler angles. Now we can compute an error quaternion that describes the necessary rotation to achieve the full desired attitude after rotating by $\mathbf{q}_{e,rp}$ as

$$\mathbf{q}_{e,y} = (\hat{\mathbf{q}} \otimes \mathbf{q}_{e,rp})^{-1} \otimes \mathbf{q}_{des}. \tag{B.20}$$

Note that the x and y components of $\mathbf{q}_{e,y}$ are always zero. Similarly to (B.15), we can now compute the desired yaw rate r_{des} from $\mathbf{q}_{e,y}^{(z)}$ with a gain p_{yaw}. Splitting the attitude control into these two parts allows us to have different gains for p_{rp} and p_{yaw}, which is desirable due to different control limits.

B.2.4 Low-Level Controller

The commands sent to the low-level controller on the PX4 are the desired body rates $\boldsymbol{\omega}_{des}$ and the desired mass-normalized thrust c_{des}. From the desired body rates $\boldsymbol{\omega}_{des}$ and the measured body rates $\tilde{\boldsymbol{\omega}}$, we can compute desired torques $\boldsymbol{\tau}_{des}$ with a feedback linearizing control scheme:

$$\boldsymbol{\tau}_{des} = \mathbf{J} \cdot \mathbf{P}_{att} \cdot (\boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}}) + \hat{\boldsymbol{\omega}} \times \mathbf{J}\hat{\boldsymbol{\omega}}, \tag{B.21}$$

where $\mathbf{P}_{att} = \text{diag}\left(p_{pq}, p_{pq}, p_r\right)$. Then, we can substitute $\boldsymbol{\tau}_{des}$ and c_{des} into (B.7) and (B.6) and solve them for the desired rotor thrusts that must be applied.

B.2.5 IMU-Based Attitude Estimation

In recovering after a loss of visual tracking, or when launching a quadrotor by hand, we need to have an attitude estimate available for controlling the attitude until the vision pipeline is initialized and running. We achieve this by using a quaternion-based attitude state estimator that fuses the measurements of the gyroscopes and accelerometers at 500 Hz. The implemented attitude estimator works in a prediction-update scheme where the prediction is performed based on the gyroscope measurements and the update step is performed based on the accelerometer measurements.

We predict the attitude estimate at the time of the current IMU measurement from the previous attitude estimate by integrating the gyroscope measurements over a time Δt between the previous and the current IMU measurement. This integration is performed with a zero-th order quaternion integration, as described in [167], assuming that the body rates are constant over a time Δt

$$\begin{aligned}\hat{\mathbf{q}}_{pred}(k) = &\left(\mathbf{I}_4 \cdot \cos\left(\frac{\|\hat{\boldsymbol{\omega}}\|\Delta t}{2}\right)\right.\\ &\left.+ \frac{2}{\|\hat{\boldsymbol{\omega}}\|} \cdot \boldsymbol{\Lambda}(\hat{\boldsymbol{\omega}}) \cdot \sin\left(\frac{\|\hat{\boldsymbol{\omega}}\|\Delta t}{2}\right)\right) \cdot \hat{\mathbf{q}}(k-1),\end{aligned} \tag{B.22}$$

where $\mathbf{I}_4 \in \mathbb{R}^{4\times4}$ denotes the identity matrix. We chose a zero-th order integration since it has a performance similar to a first order integration, but at a lower computational load. Note that (B.22) can only be evaluated if $\|\hat{\boldsymbol{\omega}}\| \neq 0$, otherwise we keep the attitude estimate prediction constant $\hat{\mathbf{q}}_{pred}(k) = \hat{\mathbf{q}}(k-1)$.

As long as the quadrotor is hand held, we assume that the accelerometers are measuring $\tilde{\mathbf{a}} = -\mathbf{g}$ on average, which gives us information about the gravity direction. Therefore, we correct the predicted attitude estimate such that the corresponding body z direction $\hat{\mathbf{e}}^B_{z,pred}$ rotates towards the measured acceleration $\tilde{\mathbf{a}}$. To do so, we compute the angle and axis of rotation required to align $\hat{\mathbf{e}}^B_{z,pred}$ with $\tilde{\mathbf{a}}$ as

$$\beta = \arccos\left({}_B\hat{\mathbf{e}}^B_{z,pred} \cdot \frac{\tilde{\mathbf{a}}}{\|\tilde{\mathbf{a}}\|}\right), \tag{B.23}$$

$${}_B\mathbf{h} = \frac{{}_B\tilde{\mathbf{a}} \times_B \hat{\mathbf{e}}^B_{z,pred}}{\|{}_B\tilde{\mathbf{a}} \times_B \hat{\mathbf{e}}^B_{z,pred}\|}. \tag{B.24}$$

Since the measured acceleration is noisy, we weigh the angle β by a gain $k_{corr} < 1$ and

use it to design the following correction quaternion:

$$\mathbf{q}_{corr} = \begin{bmatrix} \cos(k_{corr} \cdot \frac{\beta}{2}) \\ {}_B\mathbf{h} \sin(k_{corr} \cdot \frac{\beta}{2}) \end{bmatrix}, \tag{B.25}$$

which we apply to the predicted attitude estimate to get the corrected attitude estimate at the current time

$$\hat{\mathbf{q}}(k) = \hat{\mathbf{q}}_{pred}(k) \otimes \mathbf{q}_{corr}. \tag{B.26}$$

Note that the update step is only performed if the measured body rates are small, i.e. $\|\tilde{\boldsymbol{\omega}}\| < 0.5\,\mathrm{rad\,s^{-1}}$, and the magnitude of the measured acceleration is close to the magnitude of the gravitational acceleration, i.e. $|\|\tilde{\mathbf{a}}\| - g| < 1.0\,\mathrm{m\,s^{-2}}$. The first acceleration measurement that meets these criteria is used to get an initial estimate of the gravity direction.

Since this attitude estimator is based on quaternions, it is free of singularities, which is necessary because we want to be able to recover from any initial attitude. It is furthermore based on the assumption that the accelerometers are measuring $\tilde{\mathbf{a}} \approx -\mathbf{g}$ on average, which is valid when the quadrotor is hand held but not when it is flying freely. Still, in near-hover conditions, the attitude estimator does not drift in the roll and pitch estimates (see Fig. B.8) because of aerodynamic effects as described in [100].

B.2.6 Height Estimation

To estimate the vertical position z and velocity $\dot{z}$ of the quadrotor, we use a *TeraRanger One* sensor and fuse its measurements with the acceleration measurements in a fixed-gain Kalman filter. The prediction step is performed as

$$\begin{bmatrix} z_{prior} \\ \dot{z}_{prior} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{z}_k \\ \hat{\dot{z}}_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \cdot \tilde{\ddot{z}}_{imu}, \tag{B.27}$$

and the update step is performed using a fixed gain $\mathbf{K}$ as

$$\begin{bmatrix} \hat{z}_{k+1} \\ \hat{\dot{z}}_{k+1} \end{bmatrix} = \begin{bmatrix} z_{prior} \\ \dot{z}_{prior} \end{bmatrix} + \mathbf{K} \cdot \left(\tilde{z} - \hat{z}_{prior} \right). \tag{B.28}$$

B.3 Recovery and Automatic Initialization

In this section, we describe the procedure that our quadrotor executes to recover from a failure of the state estimation pipeline or a manual throw. The recovery procedure is divided into five sequential stages. We describe the control algorithm for each stage

and explain the conditions that must be met before advancing to the next stage. These conditions are chosen such that the respective controller can also satisfy them during the following stages. Each stage makes use of the controller described in Section B.2 but with different gains. The duration of each stage depends on how long it takes the controller to meet the conditions to advance to the next stage. The scheme of five stages allow to recover from various conditions after a manual throw or after a failure in the state estimation where stages 1, 2 and 4 might be skipped entirely when the conditions for the subsequent stage are already satisfied.

For our physical platform, we consider a quadrotor equipped with a monocular visual-inertial system consisting of a single camera, an IMU, and a down-looking distance sensor as described further in Section B.4.1. We demonstrate our recovery procedure on the scenario where the quadrotor is thrown in the air by hand, and automatically initializes its vision pipeline such that it can control its position purely based on a vision-based state estimate.

B.3.1 Launch Detection

As a first step to recover after tossing the quadrotor in the air, it needs to detect the launch for which it uses its accelerometers. Ideally, without disturbances and noise, the accelerometers measure $\tilde{\mathbf{a}} = -\mathbf{g}$ when standing still (e.g. on the ground) and $\tilde{\mathbf{a}} = \mathbf{0}$ when the quadrotor is in free fall. When in flight, the accelerometers ideally measure just the accelerations due to the applied rotor thrust, i.e. $\tilde{\mathbf{a}} = \mathbf{c}$. Hence, when the quadrotor is launched, we can detect a drop in the measured accelerations to a value corresponding to the currently applied thrust. Therefore, we start recovering when we measure

$$\|\tilde{a}\|_{mean} < \|c_{idle}\| + t_{launch}, \tag{B.29}$$

where $\|\tilde{a}\|_{mean}$ is the norm of the measured acceleration averaged over the last 50 ms, c_{idle} is the mass-normalized idle thrust that prevent the rotors from standing still and $t_{launch} = 2.0\,\mathrm{m\,s^{-2}}$ is a threshold parameter.

B.3.2 Recovery and Initialization Steps

Attitude Control

Immediately after a launch is detected, the quadrotor starts to control its orientation to be horizontal. To do so, we use the attitude controller described in Section B.2.3 together with an IMU based estimate of the attitude as described in Section B.2.5. Since we have no information on height and vertical velocity at this stage, we apply a fixed mass-normalized thrust equal to the gravitational acceleration $c = g$.

As soon as the distance sensor is oriented towards the ground, i.e. the angle between the body z axis $\mathbf{e}_z^B$ and the world z axis $\mathbf{e}_z^W$ (see Fig. B.3) is smaller than 20° and the roll and pitch rates are small, i.e. $\|\hat{\boldsymbol{\omega}}^{(x,y)}\| < 10\,\mathrm{rad\,s^{-1}}$, we initialize the height filter and switch to the next stage.

Attitude and Vertical Velocity Control

Once the distance sensor is pointing towards the ground, we control the horizontal velocity to zero using our position controller (B.8) but set the proportional gain $\mathbf{P}_{pos}$ and the vertical velocity gain d_{xy} to zero. The vertical velocity is estimated from the distance sensor and the IMU as described in Section B.2.6. As soon as the vertical velocity is small enough, i.e. $\|\dot{z}\| < 0.3\,\mathrm{m\,s^{-1}}$, we set the current height as the reference height and switch to the next stage.

Attitude and Height Control

Once the height controller is active, we stabilize the height relative to the ground, together with the attitude. Again, we make use of our position controller (B.8) but now only set p_{xy} and d_{xy} to zero. At this stage, due to the lack of horizontal position information, the quadrotor drifts in a horizontal plane. We use this horizontal translation to initialize the vision-based state-estimation pipeline with an initial scale corresponding to the current height estimate. After it is initialized, we switch to the next stage.

Velocity Control

At this point, the quadrotor can still have large horizontal velocities, which we want to lower before locking to the current position. In this stage, we use the position controller (B.8) but set the proportional gain p_{xy} to zero. Once the quadrotor reaches a small horizontal velocity, i.e. $\|\mathbf{v}^{(x,y)}\| < 0.2\,\mathrm{m\,s^{-1}}$, we lock to the current position and heading.

Position Control

In this last stage, we lock the quadrotor to the previously specified reference position until the quadrotor is given a new mission. For controlling the quadrotor to its reference position, we use the full control pipeline as described in Section B.2.3.

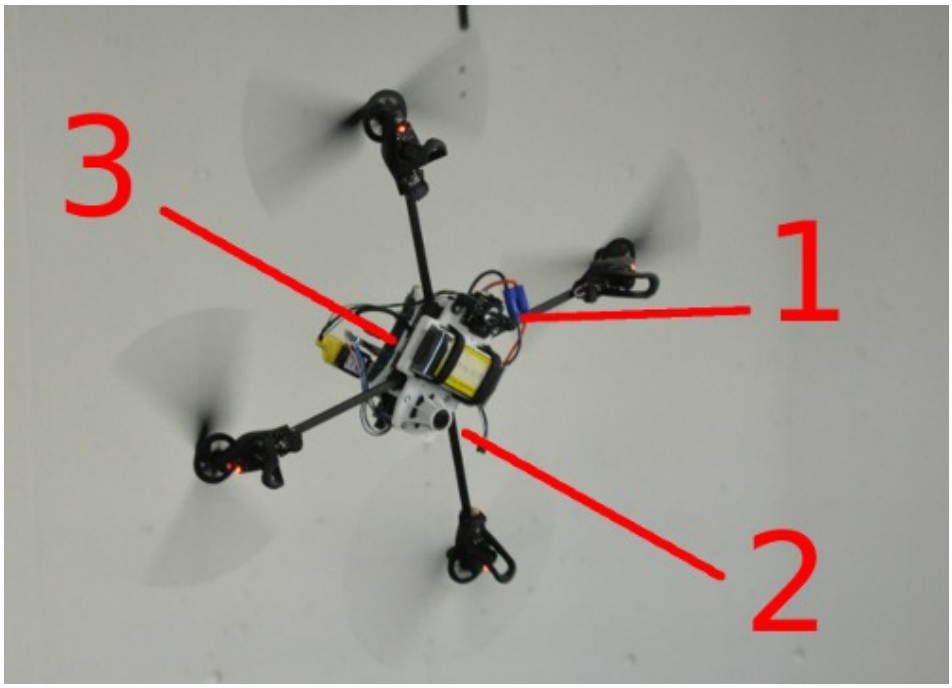


Figure B.5: A closeup of our quadrotor during recovery: 1) TeraRanger One distance sensor, 2) down-looking camera, 3) on-board electronics consisting of an Odroid U3 quad-core computer and a PX4FMU autopilot.

B.4 Experiments

B.4.1 Quadrotor Platform

We built our quadrotors from selected off-the-shelf components and custom 3D printed parts (see Fig. B.5). They rely on the frame of the Parrot AR.Drone 2.0 including their motors, motor controllers, gears, and propellers. The platform is powered by one 1,350 mA h LiPo battery which allows a flight time of approximately 10 min.

We completely replaced the electronic parts of the AR.Drone by a PX4FMU autopilot and a PX4IOAR adapter board [105]. The PX4FMU consists, among other components, of an IMU and a micro controller to read the sensors, run the low-level control (B.21), and command the motors. In addition to the PX4 autopilot, our quadrotors are equipped with an Odroid-U3 single-board computer. It contains a 1.7 GHz quad-core processor (used in Samsung smart phones) running Ubuntu 14.04 and ROS. The PX4 micro controller communicates with the Odroid board over UART (see Fig. B.2).

To stabilize the quadrotor, we make use of the gyros and accelerometers of the IMU on the PX4FMU, a downward-looking MatrixVision mvBlueFOX-MLC200w 752 $\times$ 480-pixel monochrome camera as well as a downward-looking *TeraRanger One* distance sensor. The *TeraRanger One* is an infra-red range sensor that uses time of flight to

measure distances of up to 14 m with up to 2 cm accuracy and a frequency of up to 1 kHz. It works in both indoor and outdoor environments.

The images from the camera are processed on the Odroid by means of our Semi-Direct Visual Odometry (SVO[2]) pipeline [49]. The visual-odometry pipeline outputs an unscaled pose that is then fused with the IMU readings in an Extended Kalman Filter framework (Multi Sensor Fusion (MSF) [98]) to compute a metric state estimate. More details about our quadrotor system are given in [36]

B.4.2 Indoor Experiments

To show the performance of the proposed recovery procedure, we throw a quadrotor by hand in an OptiTrack motion-capture system that we use for ground-truth comparison. For recovery, the quadrotor only uses its on-board sensors and performs all the necessary computations on board. Fig. B.6, B.7, and B.8 show the on-board state estimates compared to OptiTrack measurements. The on-board state estimates are aligned to the OptiTrack data by a single data point each. The vertical dashed lines indicate the start of the five recovery stages with corresponding numbers as described in Section B.3.2. The individual state estimates are plotted from the time on where they are used for feedback control. At the point where a state estimate becomes available, it is directly used for feedback control.

Fig. B.6 shows the height estimate from fusing TeraRanger and IMU measurements, as well as the height estimate from the visual pipeline consisting of SVO and MSF compared to the ground truth height from OptiTrack. The height estimate from fusing the TeraRanger with the IMU is used in stage 3 and 4 and the height estimate from MSF is used in stage 5 for feedback control.

Fig. B.7 shows the linear velocity estimates from MSF compared to velocity estimates from OptiTrack. In the vertical direction, we also show the velocity estimate from fusing TeraRanger and IMU measurements. This velocity estimate is used during stages 2 and 3. During stages 4 and 5, the velocity estimates from MSF are used for feedback control.

Fig. B.8 shows the roll and pitch angles from the attitude estimate purely based on the IMU measurements and from the MSF compared to the orientation measurements from OptiTrack. The IMU-based attitude estimate is used for control during stages 1, 2, and 3. During stages 4 and 5, the attitude estimate from MSF is used for feedback control. In the IMU-based attitude estimator, unit quaternions are used to represent the quadrotor's attitude and they are only transformed into Euler Angles for visualization. Note that the roll and pitch estimates from the IMU-based attitude estimator do not

[2] http://github.com/uzh-rpg/rpg_svo

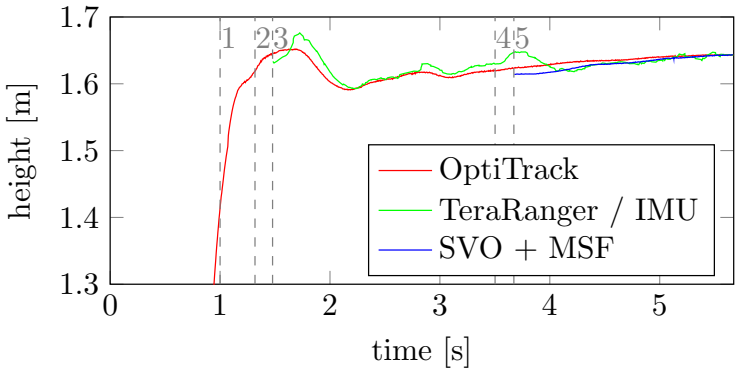


Figure B.6: Height estimates from TeraRanger / IMU fusion and the vision pipeline (SVO + MSF) compared to ground truth from OptiTrack. The vertical lines correspond to the start of each recovery stage as described in Section B.3.2.

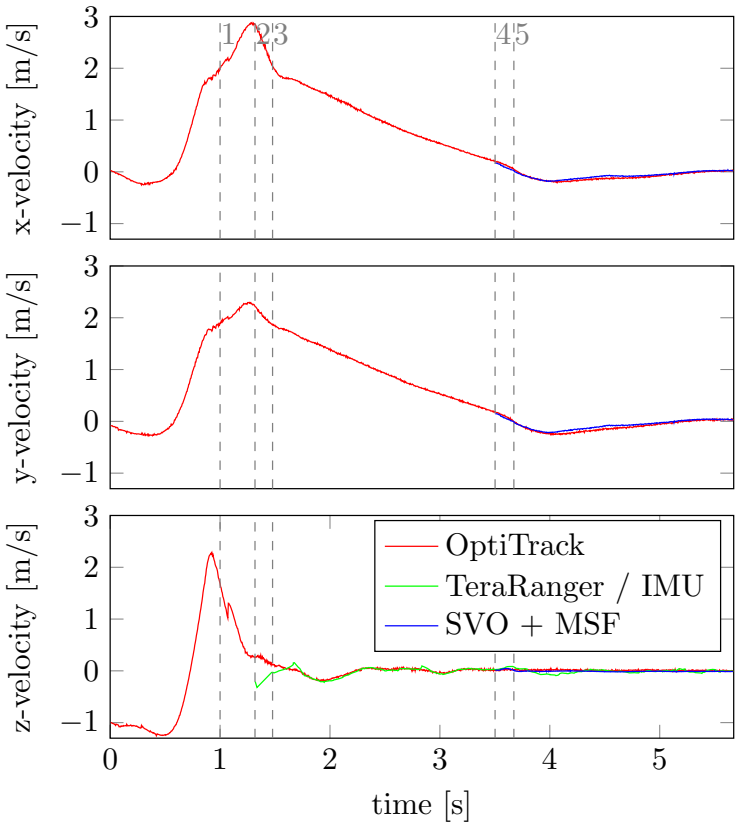


Figure B.7: Velocity estimates from the vision pipeline (SVO + MSF) and from the TeraRanger / IMU fusion compared to ground truth from OptiTrack.

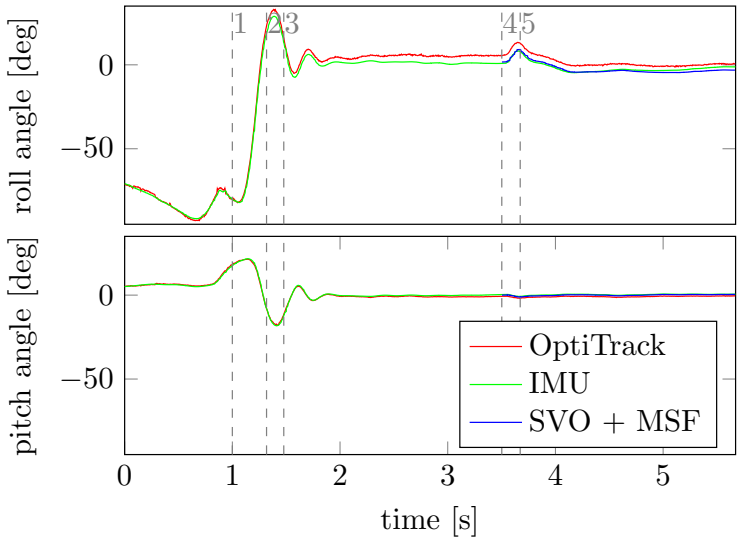


Figure B.8: Roll and Pitch estimates from the IMU-based attitude estimator (see Section B.2.5) and the vision pipeline (SVO + MSF) compared to ground truth from OptiTrack.

drift even beyond recovery.

We performed several hundred throws indoors with maximum linear accelerations above $25\,\mathrm{m\,s^{-2}}$, maximum body rates above $650\,^\circ\,\mathrm{s^{-1}}$, and maximum linear velocities exceeding $3.6\,\mathrm{m\,s^{-1}}$ with successful recoveries. We reached a success rate of more than 85 %, where failures occurred when the vision pipeline could not initialize properly, e.g., when flying close to the ground with high velocities. Examples from indoor recoveries are given in the enclosed video.

B.4.3 Outdoor Experiments

We used the same set-up as for the indoor experiments to throw the quadrotor outdoors and have it recover. Because of the absence of ground truth outdoors, plots of state estimates are not shown. In more than 30 throws with successful recovery, the quadrotor reached maximum linear accelerations above $40\,\mathrm{m\,s^{-2}}$, maximum body rates above $800\,^\circ\,\mathrm{s^{-1}}$, and maximum linear velocities above $6\,\mathrm{m\,s^{-1}}$. Examples from outdoor recoveries are given in Fig. B.1 and in the enclosed video. We achieved a similar success rate as for the indoor experiments where failures are additionally caused by disturbances of the TeraRanger due to sun light.

B.5 Conclusion

We developed a system that enables a monocular-vision-based quadrotor to recover and re-initialize its vision-based state estimation pipeline from any attitude, even with significant initial linear velocities. The on-board system receives the measurements from an IMU, a single camera, and a range sensor and fuses this information to stabilize the quadrotor by means of a cascaded control structure. We designed a recovery procedure consisting of five sequential stages with several controller types: purely inertial, range-inertial, and visual-inertial. To demonstrate its capabilities, we threw the quadrotor in the air by hand and had it recover autonomously. In contrast to existing work, the proposed system does not need any initialization before the quadrotor is launched. In indoor experiments, the state estimates obtained by our system agree well with those measured by a motion capture system. In indoor and outdoor experiments, we demonstrated that the quadrotor successfully decelerates and stabilizes within approximately two to three seconds after throwing it aggressively in the air. Our quadrotor was able to recover successfully in several hundred throws in both unknown indoor and outdoor environments with a success rate of more than 85 %. Our system not only allows instant launches but also enables mid-air re-initialization after aggressive open-loop maneuvers or in case visual tracking is lost.

C Monocular Pose Estimation

M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza. "A Monocular Pose Estimation System based on Infrared LEDs". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 907–913. DOI: 10.1109/ICRA.2014.6906962

A Monocular Pose Estimation System based on Infrared LEDs

Matthias Faessler, Elias Mueggler, Karl Schwabe and Davide Scaramuzza

Abstract — We present an accurate, efficient, and robust pose estimation system based on infrared LEDs. They are mounted on a target object and are observed by a camera that is equipped with an infrared-pass filter. The correspondences between LEDs and image detections are first determined using a combinatorial approach and then tracked using a constant-velocity model. The pose of the target object is estimated with a P3P algorithm and optimized by minimizing the reprojection error. Since the system works in the infrared spectrum, it is robust to cluttered environments and illumination changes. In a variety of experiments, we show that our system outperforms state-of-the-art approaches. Furthermore, we successfully apply our system to stabilize a quadrotor both indoors and outdoors under challenging conditions. We release our implementation as open-source software.

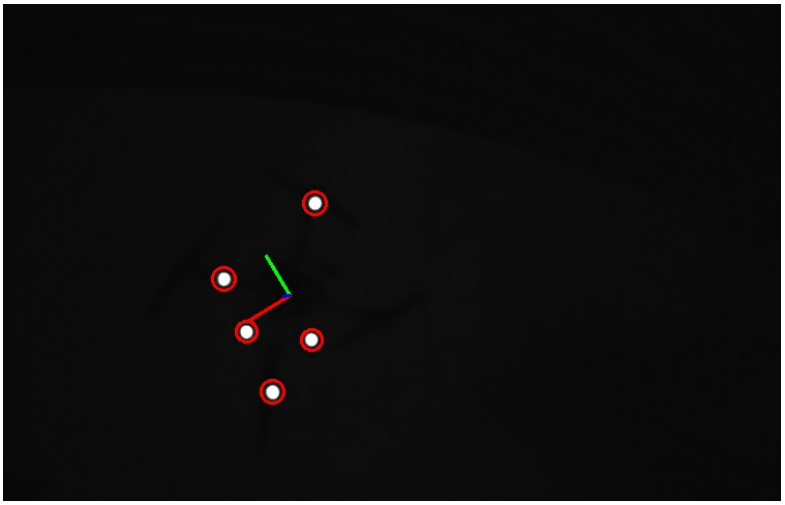
C.1 Introduction

C.1.1 Motivation

Rescue missions in disaster sites are extremely challenging for robots since they have to deal with unforeseeable and unstructured environments. Furthermore, these robots need to have many attributes, such as being able to overcome obstacles, being reasonably fast, and being able to manipulate their environment. We plan on using a team of heterogeneous robots (aerial and ground) to collaborate and make use of their individual strengths to address these challenges. Since mutual localization is one of the most fundamental parts in controlling a team of mobile robots, a system that can accurately and reliably estimate the mutual pose of the robots is necessary. For both indoor and

outdoor operations, it needs to be robust to cluttered environments, dynamic scenes, and illumination changes. Part of our robot team are quadrotors of approximately 0.5 m in size (see Fig. C.1a). Small quadrotors have fast dynamics and, thus, need a frequent and precise estimate of their 6 DOF pose to be stabilized. Furthermore, small quadrotors have limited payload and battery power, as well as limited onboard-processing power. Hence, the pose estimation system must be lightweight, energy efficient, and computationally inexpensive. Existing systems lack the combination of all these requirements.

(a) Stabilizing a quadrotor above a ground robot.



(b) View from the camera on the ground robot.

Figure C.1: A camera with an infrared-pass filter is mounted on a ground robot and used to stabilize a quadrotor above it. The red circles in (b) indicate LED detections. The pose estimate is illustrated by the projection of the body-fixed coordinate frame of the quadrotor.

We propose a pose estimation system that consists of multiple infrared LEDs and a camera with an infrared-pass filter. The LEDs are attached to the robot that we want to track, while the observing robot is equipped with the camera. Since this system operates in the infrared spectrum, the LEDs are easy to detect in the camera image. This also applies to situations with cluttered backgrounds and illumination changes. The infrared LEDs can be detected by motion-capture systems and their position on the target object can, thus, be precisely determined. Furthermore, the camera only requires a short exposure time, which allows for high frame rates (up to 90 fps in our system). To track an object, only a few LEDs need to be mounted. However, this is not an issue in terms of power consumption or payload, even for small quadrotors.

In our experiments, we compare the performance of our system to a previous approach [18] and pose estimation from AprilTags [126] as well as a motion capture system (we use OptiTrack[1]). Furthermore, we show that our system meets all the stated requirements and can be used to stabilize a quadrotor.

C.1.2 Related Work

Our work is in line with other monocular vision-based pose estimation systems [18, 26], while improving on accuracy, versatility, and performance. Since [18] uses passive markers or LEDs in the visible spectrum, their performance decreases in cluttered environments and in low-light conditions. While their system is restricted to four markers, our algorithm can handle any number of LEDs to increase robustness. Additionally, our system is robust to false detections. The setup of [26] uses a special event-based camera [92] with LEDs that blink at different frequencies. The great advantage of this camera is that it can track frequencies up to several kilohertz. Therefore, pose estimation can be performed with very low latencies. Its precision, however, is limited due to the low sensor resolution (128x128 pixels).

Nowadays, artificial patterns, such as ARTags [43] and AprilTags [126], are often used for mutual localization. In addition to a pose estimate, they also provide a unique ID of the tag. Those patterns require a large, flat area on the target object. This makes them unsuitable for micro-aerial vehicles, since it would interfere with their aerodynamics.

Another popular method for pose estimation are motion-capture systems, such as OptiTrack and Vicon.[2] While these systems yield high precision at high frame rates (up to 350 Hz), they are proprietary, expensive, and typically require a fixed installation with many cameras. In single-camera mode, OptiTrack uses the marker size, d, for estimating its 3D position and can, thus, compute a 6 DOF pose using only three markers. However, the marker size in the image, d^*, degrades quickly for larger distances to the camera

[1] http://www.naturalpoint.com/optitrack/
[2] http://www.vicon.com/

z. Consequently, also the accuracy of the pose estimate degrades: $d^* \propto d/z$. Large markers are also not suitable for micro-aerial vehicles. Nonetheless, all findings of this paper could also be applied for single-camera motion capture systems.

Estimating the camera pose from a set of 3D-to-2D point correspondences is known as Perspective from n Points (PnP) (or resection) [44]. As shown in [44], the minimal case involves three 3D-to-2D correspondences. This is called Perspective from 3 Points (P3P) and returns four solutions that can be disambiguated using one or more additional point correspondences. To solve the P3P problem, we use the algorithm in [82], which proved to be accurate while being much faster than any previous implementation. A comprehensive overview of PnP algorithms can also be found in [82] and references therein. Furthermore, we use P3P to initialize an optimization step that refines the pose by minimizing the reprojection error based on all detected LEDs.

A heuristic approach that provides near-optimal marker configurations on the target object is presented in [132]. Since the geometry of micro-aerial vehicles restricts the configuration space drastically, we do not apply such algorithms and rely on some heuristics mentioned in Section C.2.1.

The remainder of the paper is organized as follows. In Section C.2, we describe the prerequisites of our system. Our algorithm is described in Section C.3 and evaluated in Section C.4.

C.2 System Prerequisites

C.2.1 Hardware

Our system consists of infrared LEDs at known positions on the target object and an external camera with an infrared-pass filter. With at least four LEDs on the target object and the corresponding detections in the camera image, we can compute the 6 DOF pose of the target object with respect to the camera. However, to increase robustness, our system can also handle more than four LEDs on the target object. Furthermore, in case of self-occlusions or false positive detections, e.g. caused by reflections, we are still able to recover the full pose if at least four LEDs are detected.

The placement of the LEDs on the target object is arbitrary, but must be non-symmetric. In addition, the LEDs should not lie in a plane to reduce ambiguities of the pose estimation. To increase precision, they should span a large volume. Robustness can be increased if the LEDs are visible from as many view points as possible.

C.2.2 Calibration

As mentioned above, our system requires knowledge of the LED configuration, i.e. the positions of the LEDs in the reference frame of the target object. Since infrared LEDs are detectable by a motion capture system, we can use it to determine the positions of the LEDs with sub-millimeter accuracy. To do so, we first assign the desired coordinate frame to the target object in the motion capture system (we used OptiTrack) using a calibration stand (see Fig. C.2). This can be achieved by knowing the exact marker positions on the calibration stand and mounting the target object on it. Then, we can track the target object in the motion capture system and read out the positions of the single LEDs, which can be transformed into the target-object coordinate frame. Furthermore, we need to know the intrinsic camera parameters, which we obtain using the camera calibration tools of ROS.[3]

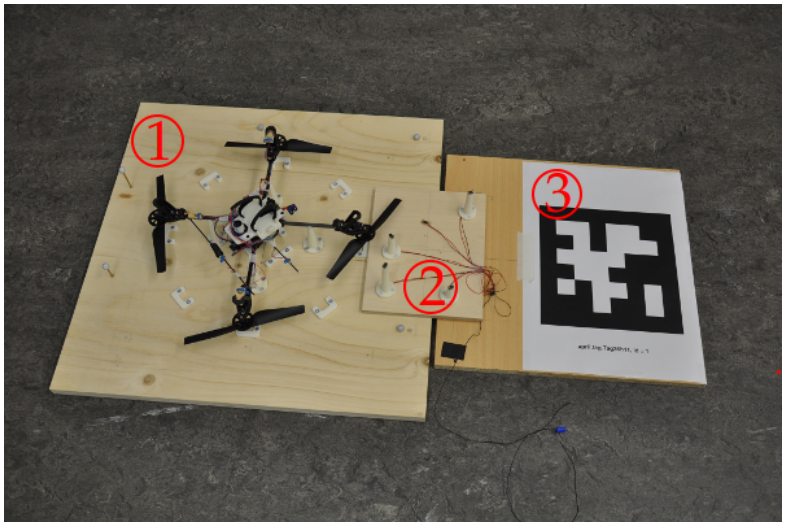


Figure C.2: Calibration stand ① to determine the exact location of the LEDs in the reference frame of a target object using a motion-capture system. The quadrotor, the target object ② and the AprilTag ③ were used to perform the experiments in Section C.4.

C.3 Algorithm

C.3.1 Overview

The flowchart of our algorithm is presented in Fig. C.3. The current camera image, the LED configuration, and previously estimated poses serve as inputs to our algorithm. In a first step, we detect the LEDs in the image. Then, we determine the correspondences using prediction or, if that fails, using a combinatorial brute-force approach. Finally, the pose is optimized such that the reprojection error of all detected LEDs is minimized.

[3] http://wiki.ros.org/camera_calibration/

This optimization also returns the covariance of the pose estimate, which is crucial information in further processing, e.g., in filtering or SLAM applications. All steps are described in more detail below.

C.3.2 Notation

We denote the LED positions on the target object as $\mathbf{l}_i \in \mathbb{R}^3$, the number of LEDs as $n_\mathcal{L}$, and the LED configuration as $\mathcal{L} = \{\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_{n_\mathcal{L}}\}$. The detections of the LEDs in the image are denoted as $\mathbf{d}_j \in \mathbb{R}^2$, measured in pixels. The number of detections is $n_\mathcal{D}$ and the set of detections is $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{n_\mathcal{D}}\}$. Note, while $\mathcal{L}$ results from the calibration, $\mathcal{D}$ depends on the current image. A correspondence of an LED $\mathbf{l}_i$ and a detection $\mathbf{d}_j$ is denoted as $\mathbf{c}_k = \langle \mathbf{l}_i, \mathbf{d}_j \rangle \in \mathcal{C} \subset \mathcal{L} \times \mathcal{D}$. Poses are denoted as $P \in SE(3)$. We use grayscale images $\mathbf{I}(u, v) : \mathbb{N}^{w \times h} \to \{0, 1, \ldots, 255\}$, where w and h denote the image width and height, respectively.

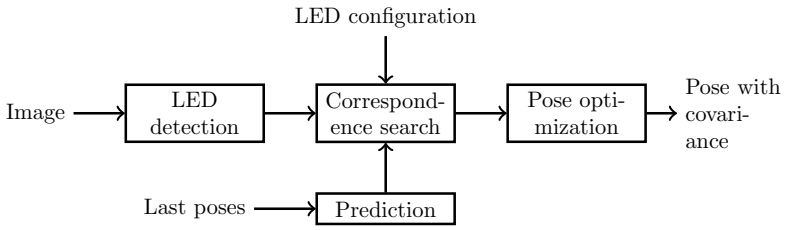


Figure C.3: Flowchart showing the main steps of our algorithm.

C.3.3 LED Detection

Since we are using infrared LEDs whose wavelength matches the infrared-pass filter in the camera, they appear very bright in the image compared to their environment. Thus, a thresholding function is sufficient to detect the LEDs $\mathcal{D}$,

$$\mathbf{I}'(u, v) = \begin{cases} \mathbf{I}(u, v), & \text{if } \mathbf{I}(u, v) > \text{threshold}, \\ 0, & \text{otherwise.} \end{cases} \tag{C.1}$$

This threshold parameter depends on the shutter speed of the camera settings. However, we found that a large range of parameters works well (80–180). We then apply Gaussian smoothing and group neighboring pixels to blobs. To estimate the center of these blobs with sub-pixel accuracy, we weigh the pixels with their intensity. The center is then calculated using first image moments that are defined as

$$M_{pq} = \sum_u \sum_v u^p v^q \mathbf{I}'(u, v). \tag{C.2}$$

The weighted center, i.e. the (distorted) LED detection in the image, is then

$$\hat{u} = M_{10}/M_{00}, \tag{C.3}$$
$$\hat{v} = M_{01}/M_{00}. \tag{C.4}$$

In all calculations to come, we assume the standard pinhole camera model. Thus, we have to correct the detections $\mathbf{d}_j$ for radial and tangential distortion. We do this using the OpenCV library [17].

C.3.4 Correspondence Search

Since the different LEDs cannot be distinguished from each other in the image, we need to find the correspondences between the LED detections, $\mathcal{D}$, in the image and the LEDs, $\mathcal{L}$, on the target object. To do so, we make use of the P3P algorithm in [82] to compute four pose candidates for every combination of three detections in the image, $\mathcal{D}_3$, and every permutation of three LEDs on the target object, $\mathcal{L}_3$. For every pose candidate, we then project the LEDs that were *not* used to compute the pose candidate, $\mathcal{L} \setminus \mathcal{L}_3$, into the camera image. If such a reprojection has a nearest neighbor of the detections $\mathcal{D}$ closer than a threshold λ_r, we consider the LED to correspond to this detection. For the reprojection-distance threshold, we typically use $\lambda_r = 5\,\text{pixels}$. To be robust to outliers, we form a histogram with bins for every detection-LED pair. A histogram bin is increased whenever a pair is considered to be a correspondence. This procedure returns the set of correspondences $\mathcal{C}$ and is summarized in Algorithm 1. The procedure for finding the final correspondences from the histogram is illustrated in Fig. C.5.

For $n_\mathcal{D}$ detections and $n_\mathcal{L}$ LEDs on the object, we will obtain N pose candidates,

$$N = 4 \cdot \binom{n_\mathcal{D}}{3} \cdot \frac{n_\mathcal{L}!}{(n_\mathcal{L} - 3)!}. \tag{C.5}$$

This number grows quickly for a large $n_\mathcal{D}$ or $n_\mathcal{L}$. However, since we use only a few LEDs (typically four or five) and false-positive detections are rare, this is not an issue. Numbers of pose candidates computed according to (C.5) are shown in Fig. C.4.

C.3.5 Prediction

Since the brute-force matching in the previous section can become computationally expensive, we predict the next pose using the current and the previous pose estimates. A constant-velocity model is used for prediction. The pose P is parametrized by twist

$n_\mathcal{L}$ \ $n_\mathcal{D}$	4	5	6	7	8
4	384	960	1,920	3,360	5,376
5	960	2,400	4,800	8,400	13,440
6	1,920	4,800	9,600	16,800	26,880

Figure C.4: Number of pose candidates N based on the number of detections $n_\mathcal{D}$ and the number of LEDs on the target object $n_\mathcal{L}$.

Algorithm 1 Correspondence search

```
for all D_3 ∈ Combinations(D, 3) do
  for all L_3 ∈ Permutations(L, 3) do
    L_r ← L \ L_3
    P ← P3P(D_3, L_3)
    for all P ∈ P do
      found ← False
      for all l ∈ L_r do
        p ← project(l, P)
        for all d ∈ D do
          if ||d − p||^2 < threshold then
            inc(histogram(l, d))
            found ← True
      if found then
        inc(histogram(L_3, D_3))
```

coordinates $\boldsymbol{\xi}$. We predict the next pose linearly [54, p. 511],

$$\hat{\boldsymbol{\xi}}_{k+1} = \boldsymbol{\xi}_k + \Delta T \left(\boldsymbol{\xi}_k - \boldsymbol{\xi}_{k-1} \right), \tag{C.6}$$

$$\Delta T = \begin{cases} 0, & \text{if } n_P = 1, \\ (T_{k+1} - T_k)/(T_k - T_{k-1}), & \text{if } n_P \geq 2, \end{cases} \tag{C.7}$$

where T_k is the time at step k and n_P the number of previously estimated poses.

Using the predicted pose, we project the LEDs into the camera image. We then match each prediction with its closest detection, if they are closer than a threshold. (Note that this threshold is different from λ_r). This condition prevents false correspondences, e.g. if an LED is not detected. We typically use 5 pixels for that threshold. We then

$\mathcal{D}$ \ $\mathcal{L}$	$\mathbf{l}_1$	$\mathbf{l}_2$	$\mathbf{l}_3$	$\mathbf{l}_4$	$\mathbf{l}_5$
$\mathbf{d}_1$	1	**12**	0	1	0
$\mathbf{d}_2$	0	3	2	1	**8**
$\mathbf{d}_3$	1	0	1	**13**	1
$\mathbf{d}_4$	1	0	1	4	1
$\mathbf{d}_5$	2	1	0	1	1
$\mathbf{d}_6$	**11**	3	0	2	2

Figure C.5: Correspondence histogram. The numbers indicate how often a small reprojection error of LED $\mathbf{l}_i$ to the detection $\mathbf{d}_j$ was found. Under ideal conditions, this value is $\binom{n_\mathcal{L}}{3}$ for a correspondence and zero otherwise. In practice, we iteratively search for the highest number in the histogram, take the respective LED and image point as the correspondence, and then ignore that column in all subsequent iterations. Note that this allows a detection to correspond to multiple LEDs, but not vice versa (cf. Section C.4.2). In this example, in the first iteration, we match $\mathbf{l}_4$ and $\mathbf{d}_3$, i.e. $\mathbf{c}_1 = \langle \mathbf{l}_4, \mathbf{d}_3 \rangle$. All further correspondences are also marked in bold. Note that $\mathbf{l}_3$ was not matched since all remaining entries in its column are lower than a threshold (we chose $0.5\binom{n_\mathcal{L}}{3}$). This LED might be occluded (cf. Section C.4.2) or its detection failed.

check if the predicted correspondences are correct. To do so, we compute the four pose candidates with the P3P algorithm for every combination of three correspondences. We then compute the projection of the remaining LEDs and check if at least 75 % of them are below the reprojection threshold λ_r. If this is true for one of the four pose candidates of more than 70 % of the combinations of correspondences, we consider them as correct. In case we could not find the correct correspondences, we reinitialize the tracking using the brute-force method from the previous section.

C.3.6 Pose Optimization

To estimate the target-object pose, P^*, we use all correspondences in $\mathcal{C}$ and iteratively refine the reprojection error [163, p. 286f.] starting with a solution from the P3P algorithm as an initial estimate, that is

$$P^* = \arg\min_P \sum_{\langle \mathbf{l}, \mathbf{d} \rangle \in \mathcal{C}} \| \pi(\mathbf{l}, P) - \mathbf{d} \|^2, \tag{C.8}$$

where $\pi : \mathbb{R}^3 \times SE(3) \to \mathbb{R}^2$ projects an LED into the camera image. For the optimization, we parametrize the pose using the exponential map and apply a Gauss-Newton minimization scheme.

The covariance of the final pose estimate, $\mathbf{\Sigma}_P \in \mathbb{R}^{6\times6}$, is a byproduct of the Gauss-Newton scheme, since it requires the computation of the Jacobian matrix, $\mathbf{J} \in \mathbb{R}^{2\times6}$.

Using the derivation of [33, p. 182ff.], we can compute $\mathbf{J}$ in closed-form. The covariance of the pose, $\mathbf{\Sigma}_P$, is then obtained by [13]

$$\mathbf{\Sigma}_P = \left(\mathbf{J}^\top \mathbf{\Sigma}_\mathcal{D}^{-1} \mathbf{J}\right)^{-1}, \tag{C.9}$$

where $\mathbf{\Sigma}_\mathcal{D} \in \mathbb{R}^{2\times 2}$ is the covariance of the LED detections, which we conservatively set to $\mathbf{\Sigma}_\mathcal{D} = \mathbf{I}_{2\times 2} \cdot 1\,\text{pixel}^2$.

C.4 Evaluation

C.4.1 Benchmarks

To evaluate our system, we compare it to a previous system [18] and to AprilTags [126]. A MatrixVision mvBlueFOX-MLC200w monochrome camera[4] with an infrared-pass filter, a resolution of 752x480 pixels, and a field of view of 90° was used for the experiments. Furthermore, we added reflective markers to the camera to obtain ground truth in an OptiTrack motion-capture system. On the target object, we mounted SMD LEDs (of type Harvatek HT-260IRPJ or similar) since they proved to have a wide radiation pattern. We used either a configuration of four or five infrared LEDs on the target object (see Fig. C.2). Both configurations have a circumsphere radius of 10.9 cm. Since the infrared LEDs are directly visible in the motion capture system, no additional markers were needed to obtain the ground truth data of the target object. To have a direct comparison, we attached an AprilTag with edge length of 23.8 cm to the target object. For pose estimation from the AprilTags, we used a C++ implementation.[5]

In a first run, the target object is positioned at a fixed location while the camera is moving. We used $n_\mathcal{L} = 4$ LEDs on the target object and performed excitations of the camera in all six degrees of freedom. Fig. C.6 shows position and orientation as well as the respective errors. Since our setup is virtually identical to [18] and the trajectory follows the similar excitations in all six degrees of freedom, we claim that the results are comparable. In Table C.1, we compare our performance to the system in [18] and to AprilTags [126]. As an orientation error metric, we used the angle of the angle-axis representation. Since we cannot measure the precise location of the center of projection of the camera, we use the first 10 % of the data for hand-eye calibration. We also estimate the pose of the AprilTag with respect to the target object in the same way. The dataset consists of 7,273 images. In 2 images (0.03 %), not all 4 LEDs could be detected. In another 2 images, no solution was found. Thus, in 99.94 % of all images, a good estimate was found.

In a second experiment, we evaluated the error with respect to the distance between

[4] http://www.matrix-vision.com/
[5] http://people.csail.mit.edu/kaess/apriltags/

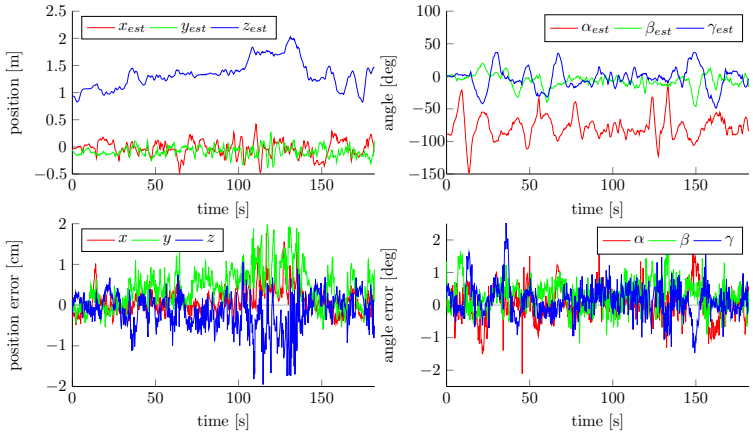
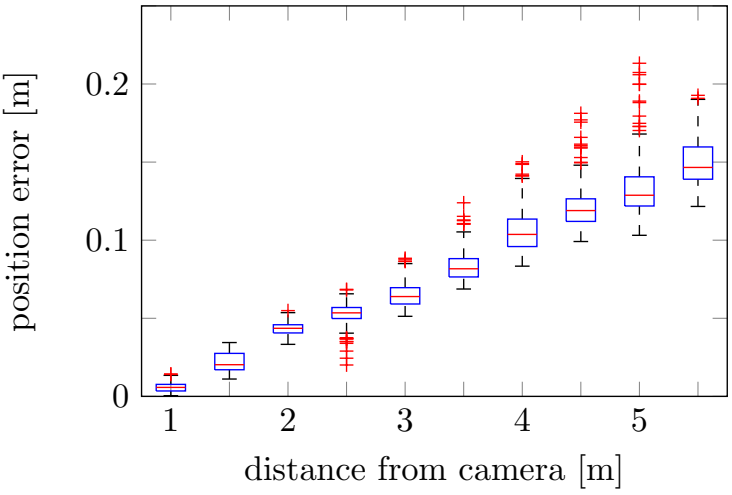


Figure C.6: Estimation of position and orientation, as well as the respective errors. Ground truth is not shown because there is no visible difference to the estimated values at this scale. The orientation is parametrized with Euler angles, i.e. yaw (α), pitch (β), and roll (γ). The target object was equipped with $n_{\mathcal{L}} = 4$ LEDs. For four out of a total of 7,273 images, no estimate could be resolved.

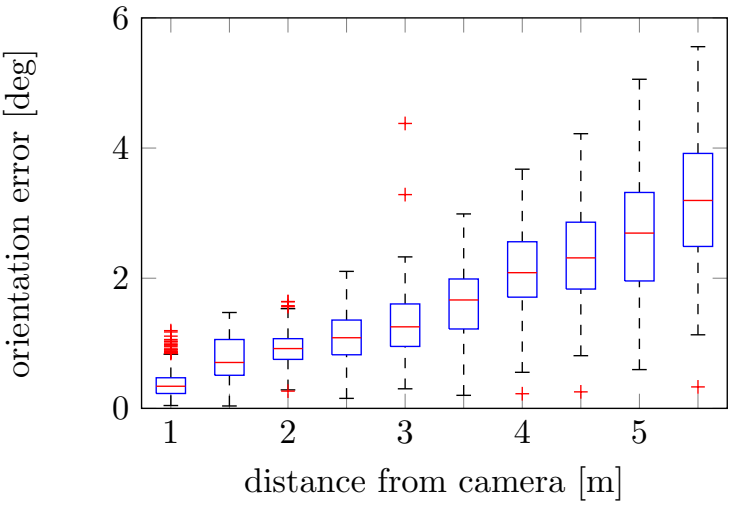
Table C.1: Comparison of pose estimation performance.

	April-Tags [126]	Breitenmoser et al. [18]	Our system	
Mean Position Error	1.41	1.5	**0.74**	cm
Standard Deviation	1.02	0.7	**0.46**	cm
Max Position Error	11.2	12.1	**3.28**	cm
Mean Orientation Error	1.53	1.2	**0.79**	°
Standard Deviation	1.61	**0.4**	0.41	°
Max Orientation Error	19.5	4.5	**3.37**	°

the camera and the target object. We used $n_{\mathcal{L}} = 5$ LEDs on the target object to increase robustness. The camera was moved from 0.8 m to 5.6 m, while recording a total of 2,651 images. Fig. C.7 shows the boxplots for both position and orientation. For the orientation error, we used again the axis-angle representation. In 3 images (0.04 %) at more than 5 m distance, incorrect correspondences lead to pose estimates that were off by more than 90° in orientation. We consider them as outliers and, thus, they are not shown in Fig. C.7b.

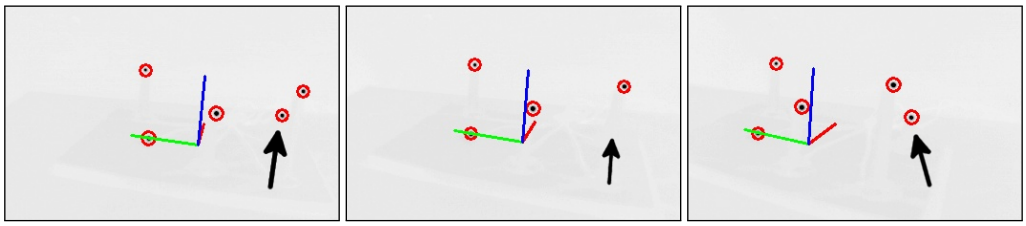


(a) Position error.



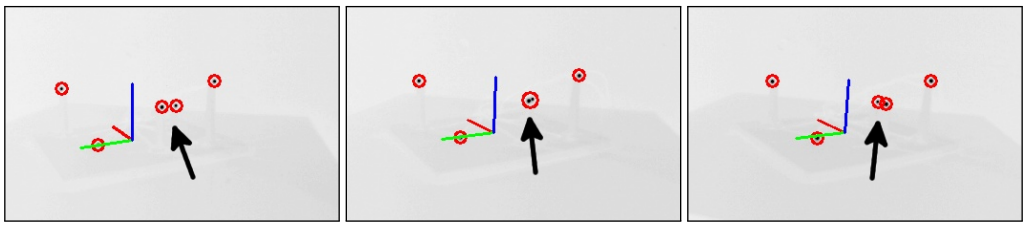
(b) Orientation error.

Figure C.7: Boxplot of the pose estimation errors with respect to the distance between the target object and the camera. The target object was equipped with $n_{\mathcal{L}} = 5$ LEDs.



(a) T_1 **(b)** T_2 **(c)** T_3

Figure C.8: Camera images (a) before, (b) during, and (c) after an LED occlusion. The camera images were inverted for better contrast. The estimation errors for this experiment are shown in Fig. C.10. The arrow indicates the LED that is occluded.



(a) T_1 **(b)** T_2 **(c)** T_3

Figure C.9: Camera images (a) before, (b) during, and (c) after an alignment of two LEDs. The camera images were inverted for better contrast. The estimation errors for this experiment are shown in Fig. C.11. The arrow indicates the two LEDs that were aligned.

C.4.2 Occlusions and Alignment of LEDs

Here we take a deeper look at two special cases. First, we evaluate the estimation error in situations where an occlusion occurs. In Fig. C.8b, we show such a situation. Since at least four LEDs are always visible for the entire duration of the occlusion, the estimation error does not change significantly (cf. Fig. C.10).

Secondly, we look at the situations where two LEDs appear as one in the image (e.g. in Fig. C.9b). In such situations, they cannot be detected separately. Thus, as soon as the two LEDs are detected as one, there is an immediate increase in the estimation error. As the LEDs appear closer to each other, the error decreases until the two LEDs are almost perfectly aligned. It then increases until the two LEDs can again be detected separately, whereafter it drops to the initial values. This behavior can be seen in Fig. C.11.

C.4.3 Quadrotor Stabilization

To show the applicability of our system in a real-world scenario, we demonstrate closed-loop control of a quadrotor[6] using pose estimates of our system at 40 Hz. We

[6] A video is included as an attachment to this paper.

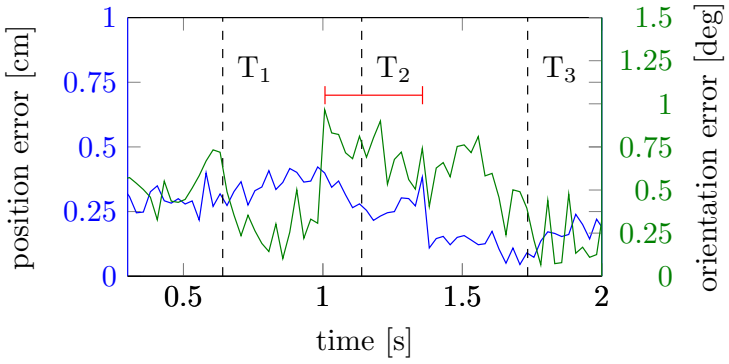


Figure C.10: Error plots of position and orientation during an LED occlusion. As an orientation error metric, we used the angle of the angle-axis representation. The red interval indicates the duration of the occlusion. The times T_i correspond to the images in Fig. C.8.

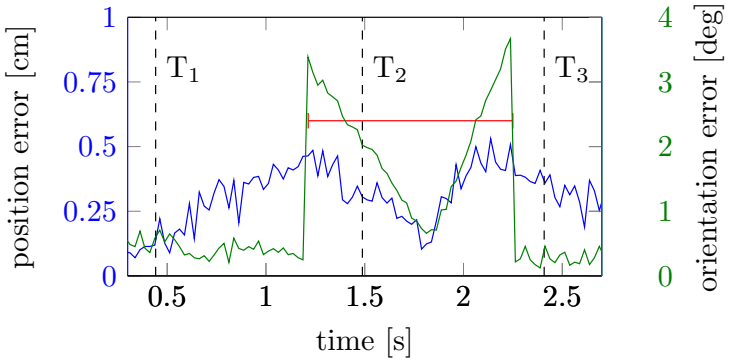


Figure C.11: Error plots of position and orientation when two LEDs appear as one in the camera image. As an orientation error metric, we used the angle of the angle-axis representation. The red interval marks the duration of the alignment. The times T_i correspond to the images in Fig. C.9.

attached $n_{\mathcal{L}} = 5$ LEDs to a quadrotor, which is based on the PIXHAWK platform [105] (see Fig. C.2), and mounted the camera on a KUKA youBot [14] (see Fig. C.1). We used a lens with field of view of 120°. Our system is robust enough to handle illumination changes from daylight to complete darkness, false detections, occluded LEDs, and dynamic backgrounds. It is also fast and precise enough to stabilize the quadrotor when it gets pushed or flies outdoors with unpredictable winds.

C.4.4 Execution Time

The mean and maximum execution times for each step of our algorithm can be found in Table C.2. They were measured while running our system on a dataset with 2,400 images and LED configurations consisting of 4 and 5 LEDs. For the timing, we enforced

Table C.2: Execution times of the individual steps of our algorithm (corresponding to subsections C.3.3 to C.3.6)

Number of LEDs		LED detection [ms]	Correspondence search [ms]	Prediction [ms]	Pose optimization [µs]	Total (w/o prediction) [ms]	**Total (w/ prediction)** [ms]
$n_{\mathcal{L}} = 4$	Mean	2.7	1.1	0.2	31	5.0	**3.8**
	σ	0.9	0.6	0.1	10	1.4	**1.1**
	Maximum	5.1	5.7	0.6	94	10.1	**6.5**
$n_{\mathcal{L}} = 5$	Mean	2.7	4.9	0.3	36	9.0	**3.8**
	σ	0.8	2.0	0.1	11	2.5	**1.1**
	Maximum	5.1	14.0	0.7	93	17.6	**9.3**

a brute-force correspondence search in each step. However, if we use prediction, this search is required less than 0.2 % of the time. We used a laptop with an Intel i7-3720 (2.60 GHz) processor. Note that on average the LED detection makes up 71 % of the execution time. This could be drastically reduced by defining a region of interest around the predicted detections, as is done in [18].

C.5 Conclusions

We presented an accurate, versatile, and robust monocular pose tracking system based on infrared LEDs. Comprehensive experiments showed its superiority over previous approaches for pose estimation and its applicability to robots with fast dynamics such as quadrotors. Our system is available as an open-source ROS [136] package, so that it can easily be integrated into other robotic platforms.

Future work will include the extension to track multiple objects and will integrate the dynamical model of the target object for prediction and filtering. Furthermore, we plan to use the system for mutual localization in a team of quadrotors.

D Aerial and Ground Robot Collaboration

E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza. "Aerial-guided Navigation of a Ground Robot among Movable Obstacles". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. 2014, pp. 1–8. DOI: 10.1109/SSRR.2014.7017662

Aerial-guided Navigation of a Ground Robot among Movable Obstacles

Elias Mueggler, Matthias Faessler, Flavio Fontana, and Davide Scaramuzza

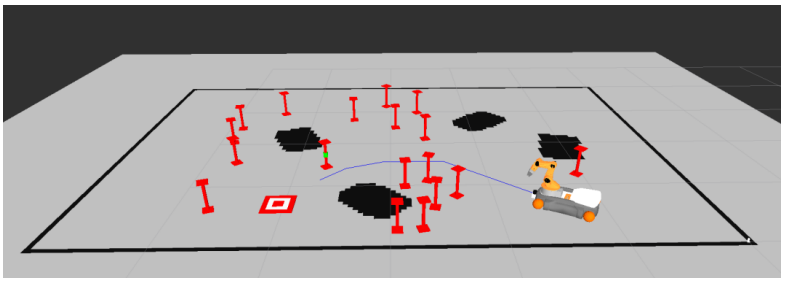
Abstract — We demonstrate the fully autonomous collaboration of an aerial and a ground robot in a mock-up disaster scenario. Within this collaboration, we make use of the individual capabilities and strengths of both robots. The aerial robot first maps an area of interest, then it computes the fastest mission for the ground robot to reach a spotted victim and deliver a first-aid kit. Such a mission includes driving and removing obstacles in the way while being constantly monitored and commanded by the aerial robot. Our mission-planning algorithm distinguishes between movable and fixed obstacles and considers both the time for driving and removing obstacles. The entire mission is executed without any human interaction once the aerial robot is launched and requires a minimal amount of communication between the robots. We describe both the hardware and software of our system and detail our mission-planning algorithm. We present exhaustive results of both simulation and real experiments. Our system was successfully demonstrated more than 20 times at a trade fair.

D.1 Introduction

Since 2001, rescue robots have been deployed for disaster response and have been used at least 29 times to date [120]. For example, after the earthquake and tsunami in Fukushima, Japan in 2011, ground robots were utilized to explore the situation in the contaminated reactor building [121].

In all previous disaster response missions, the robots were remote-controlled by trained professionals. Three operators per robot were required on average and the executed

(a) Our robots operating in a mock-up disaster site



(b) Corresponding mission plan

Figure D.1: After mapping the area, the aerial robot is guiding the ground robot to the goal location. All paths are blocked by obstacles, some of which can be removed by the ground robot.

missions took very long (cf. [120]). Since time is the most critical factor in rescue missions, we propose to deploy teams of *heterogeneous* robots, namely ground and aerial robots, to speed up disaster response. Their sense-act capabilities are complementary: ground robots can carry high payloads and manipulators. However, their field of view is limited and they can be blocked by obstacles on the ground. Aerial robots, in contrast, can overcome obstacles with ease and can provide a bird's-eye view, which is ideal for mapping and monitoring tasks.

To reduce the number of required operators and speed up their mission, the robots must expose a good level of autonomy. Instead of sending low-level commands, they must be able to autonomously execute high-level tasks such as "grasp that object" or "fly to location X". This allows the operator to focus on the mission instead of low-level robot details. For reliable local navigation, robots must rely only on their onboard sensors, since the communication infrastructure is likely to be affected during disaster situations.

In this work, we demonstrate the benefits of a team of heterogeneous robots in a mock-

up search-and-rescue scenario (see Figure D.1a): a victim in an unknown environment must be found and provided with a first-aid kit. We first deploy an aerial robot that maps the environment and searches for the victim. Then, a mission is planned for a ground robot to reach the victim and provide it with a first-aid kit as fast as possible. Both the time required for driving and rubble removal are considered to plan the mission. Finally, the aerial robot guides the ground robot along the computed path to the victim.

The remainder of this paper is organized as follows. In Section D.2, we review related work both on collaboration of aerial and ground robots and on Navigation Among Movable Obstacles (NAMO). Our robots and the mock-up disaster scenario are introduced in Section D.3. In Sections D.4, D.5, and D.6 we detail the mapping, planning, and execution of the missions, respectively. Finally, we evaluate the performance of our mission planning algorithm and the entire system in Section D.7.

D.2 Related Work

We first review related work on collaboration of ground and aerial robots in the search-and-rescue context. Then we provide a literature overview on Navigation Among Movable Obstacles (NAMO).

D.2.1 Collaboration of Aerial and Ground Robots

After the 2012 Mirandola earthquake in Italy, aerial and ground robots were deployed to build 3D maps of the interior of damaged buildings [85]. The robots were remote controlled and high stress and cognitive overload of both the aerial and ground robot operators were reported.

In [111], collaborative mapping of a damaged building after the 2011 Tokuhu earthquake with a ground and an aerial robot was presented. First, the ground robot was manually controlled through the building creating a 3D voxel grid. Second, locations inaccessible for the ground robot were mapped by the aerial robot.

Teams of aerial and ground robots for monitoring and tracking tasks were presented in [70]. Aerial robots search for targets which are then verified and tracked by ground robots. Only limited, high-level user interaction was required. However, the main focus was on an ad-hoc wireless network that is maintained during the mission.

In [59], an aerial robot is described to support ground personnel in searching for victims. Several search strategies are discussed. The system was tested outdoors with rescue professionals.

In [56], an aerial robot is described to assist a ground robot to reach a goal location. However, only local planning was performed and the aerial robot was piloted manually.

Since virtually all robots in today's missions are remote controlled, human-robot interaction for these scenarios was investigated by several authors, e.g. [131, 87]. Also, the recently started European project SHERPA [101] focuses on the interaction and collaboration of humans with both aerial and ground robots in alpine rescue missions.

Our work differs from those mentioned above in that we do not only map the environment, but also use these maps for mission planning and environment interaction. In particular, we make use of the ground robot's capability to *interact* with the environment using its robotic manipulator. In addition, our system is fully autonomous: no user interaction is required at any point after launching the aerial robot.

D.2.2 Navigation Among Movable Obstacles (NAMO)

Planning in modifiable environments was introduced in [177]. It was shown that, even for the simplest cases, the problem is NP-hard. A heuristic search algorithm was presented in [29]. More recently, Navigation Among Movable Obstacles (NAMO) [156, 157] became an active research topic in the field of humanoid robots. While very similar to our case, there is a difference in how we can manipulate obstacles. In all above-mentioned papers, the obstacles could be *pushed* by the robot. Here, contrarily, we can *lift* the obstacles and place them again at an *arbitrary* location.

D.3 System Overview

We propose a system consisting of a quadrotor equipped with a monocular camera (see Figure D.2) and a ground robot consisting of an omni-directional base and a 5-DOF manipulator (see Figure D.3). When operating together, the two robots have all the capabilities we require for the considered search-and-rescue missions. Combining all the required capabilities in one single robot would render the system impractical and less flexible.

We use a laptop as ground station to visualize the progress of the mission. On the ground station, we also run our mission-planning algorithm and send high-level commands to both robots.[1] As soon as the robots have received their high-level commands, they both navigate fully autonomously while running all the required algorithms onboard. In the following, we describe the quadrotor platform, the ground robot, and the mock-up disaster scenario in more detail.

[1] Note that this could also run onboard one of the robots, rendering the laptop unnecessary.

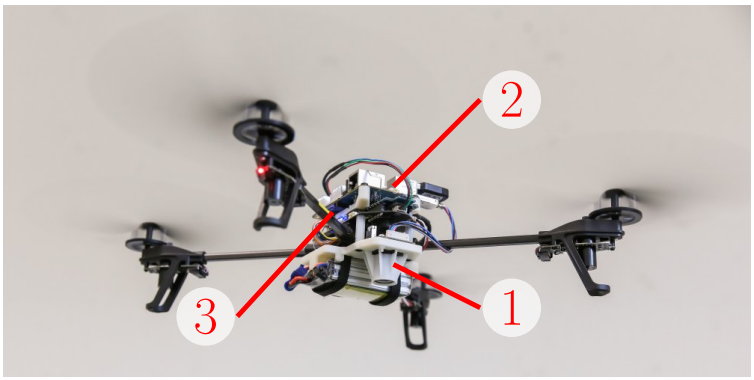


Figure D.2: A closeup of our quadrotor: down-looking camera (1), Odroid U3 quad-core computer (2), and PIXHAWK autopilot (3).

D.3.1 Aerial Robot

We built our quadrotor from selected off-the-shelf components and custom 3D-printed parts (see Figure D.2). The components were chosen according to their performance and their ability to be easily customized.

Our quadrotor relies on the frame of the Parrot AR.Drone 2.0[2] including their motors, motor controllers, gears, and propellers. The platform is powered by one 1,350 mA h LiPo battery, which allows a flight time of 10 min.

We completely replaced the electronic parts of the AR.Drone by a PX4FMU autopilot and a PX4IOAR adapter board developed in the PIXHAWK Project [105]. The PX4FMU consists, among other parts, of an IMU and a micro controller to read the sensors, run some low-level control to track desired body rates, and command the motors. Additionally to the PX4 autopilot, our quadrotors are equipped with an Odroid-U3 single-board computer.[3] It contains a 1.7 GHz quad-core processor running XUbuntu 13.10[4] and ROS.[5] The PX4 micro controller communicates with the Odroid board over UART, whereas the Odroid board communicates with the ground station over 5 GHz WiFi.

Our platform is easily reparable due to off-the-shelf components, inexpensive (1,000 USD), lightweight (below 450 g), and therefore safe to use.

To stabilize the quadrotor, we make use of the gyros and accelerometers of the IMU on the PX4FMU as well as a downward-looking MatrixVision mvBlueFOX-MLC200w 752 × 480-pixel monochrome camera.[6]

[2] http://ardrone2.parrot.com/
[3] http://www.hardkernel.com/main/products/prdt_info.php?g_code=G138745696275
[4] http://www.xubuntu.org/
[5] http://www.ros.org/
[6] http://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html

The images from the downward-looking camera are processed on the Odroid by means of our Semi-Direct Visual Odometry (SVO[7]) pipeline [49]. The visual-odometry pipeline outputs an unscaled pose which is then fused with the IMU readings in an Extended Kalman Filter framework (Multi Sensor Fusion (MSF) [98]) to compute a metric state estimate. From this state estimate and a desired trajectory, we compute the desired body rates and collective thrust, which are then sent to the low-level controller on the PX4FMU.

D.3.2 Ground Robot

We use a KUKA youBot [14] as ground robot (see Figure D.3). It consist of a mobile base and a 5-DOF manipulator with a two-finger gripper. The mobile base includes four omni-directional wheels, which allow the robot to also move sideways and rotate on the spot. This is a great advantage over standard wheels when navigating in confined spaces. Furthermore, the base is designed robust enough to carry a payload of 20 kg. The manipulator is able to lift up to 0.5 kg with its gripper. For controlling the arm and the base, the youBot comprises a mini ITX PC board with embedded Intel®Atom Dual-Core CPU running an Ubuntu operating system and ROS. To grasp objects fast and reliably, we developed a torque controller for the youBot arm [79], which allows to precisely track trajectories with its gripper.

To measure the relative position of obstacles in front of the youBot, we mounted a Hokuyo URG-04LX-UG01[8] laser scanner in front of its base. Finally, for this rescue mission, the youBot carries a first-aid kit on the side of its base as shown in Figure D.3.

D.3.3 Mock-up Disaster Site

In this work, we consider a search-and-rescue mission after a disaster where robots are sent into areas that are too dangerous to be entered by human rescuers.

As a mock-up disaster site for our experiments (see Figure D.4), we consider an area of known size (in our case 4 m $\times$ 6 m) with different types of obstacles: some of them can be removed by the ground robot ("movable obstacles") and some can only be avoided ("fixed obstacles"). Furthermore, there is a goal location that represents a victim, which has to be provided with a first-aid kit by the ground robot as fast as possible. The locations of the obstacles and the goal are not known a priori.

All obstacles, the goal, and the ground robot are marked with an AprilTag [126] such that they can easily be detected in the camera image of the flying robot (see Figure D.2). We make use of an additional AprilTag as origin tag, which provides a common

[7] http://github.com/uzh-rpg/rpg_svo
[8] http://www.hokuyo-aut.jp/02sensor/07scanner/download/products/urg-04lx-ug01/

Figure D.3: KUKA youBot equipped with an AprilTag (1), a laser scanner (2), and a first-aid kit (3).

reference frame for both robots. The origin tag is also used as reference point to define the search area that the aerial robot has to cover.

D.4 Mapping

To map the defined area, we compute a *lawn-mower pattern* covering the entire area for the flying robot to take images. Images are only taken at specified locations on the lawn-mower pattern instead of being streamed continuously in order to minimize the required communication band with. These images, together with the onboard pose estimate of the quadrotor, are sent to the ground station while the flying robot is mapping. The obstacles are then detected in the image by their attached tag. To build a metrically consistent map, we run a pose-graph optimization over all detected tags of all images. We initialize the tag poses with the estimate of the quadrotor pose and run the optimization using iSAM [75].

The detected fixed obstacles are then inserted into a grid map together with the boundaries of the defined area as illustrated in Figure D.8 in black. In addition to this grid map, we also provide the planner with the position of the movable obstacles, the pose of the ground robot, and the goal location.

D.5 Mission Planning

When planning a mission for the ground robot, we consider its position to be the center point of its base and we represent the map as a grid where we inflate all fixed obstacles

Figure D.4: Mock-up disaster site elements: fixed obstacle (1), movable obstacle (2), goal (3), and origin tag (4).

by the radius of the robot base. To grasp an obstacle, the robot's position must be on a circle around the obstacle with a radius corresponding to the distance from the gripper to the base center (see Figure D.5). We denote the points on this circle as possible grasp locations of that obstacle.

Our planning is based on the A* algorithm with the mission-execution time as cost. First, for each movable obstacle, we calculate the minimum cost to the goal location when no other movable obstacle would be present. This gives us a lower bound on the cost to go from each movable obstacle to the target location and is therefore an admissible heuristic. Then, we search for every feasible shortest path from the start location to any of the grasp locations of every movable obstacle or the goal location directly. For every path to a movable obstacle that we find, we add the cost to remove it and compute all feasible paths to all other removable obstacles or the goal location. All the possible missions that are created this way are stored in a priority queue according to their estimated cost. This process is continued until we find a feasible mission to the goal location.

Until this point, we do not consider the cost of driving the obstacle to a feasible place location, which can be necessary as described further in Section D.5.1. Therefore, all the computed costs are lower bounds and not necessarily the actual costs of the corresponding mission. When a feasible mission to the goal is found, we compute the precise cost including possible driving backwards to place obstacles. Note that this can only increase the cost of the mission. If the actual cost of the considered mission is now higher than the estimated cost of other missions in the priority queue, we have to continue the planning steps for these missions as described above. The time-optimal mission is found if its actual cost is smaller than the estimated cost of every other mission in the priority queue. This procedure is summarized in Algorithm 2.

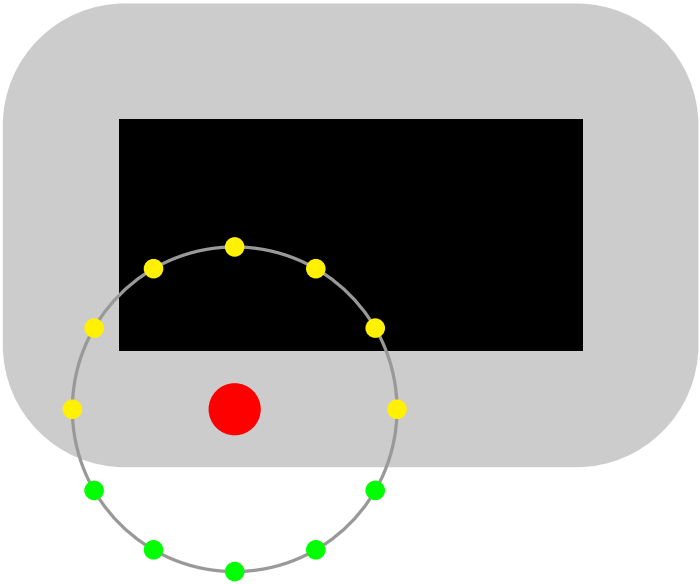
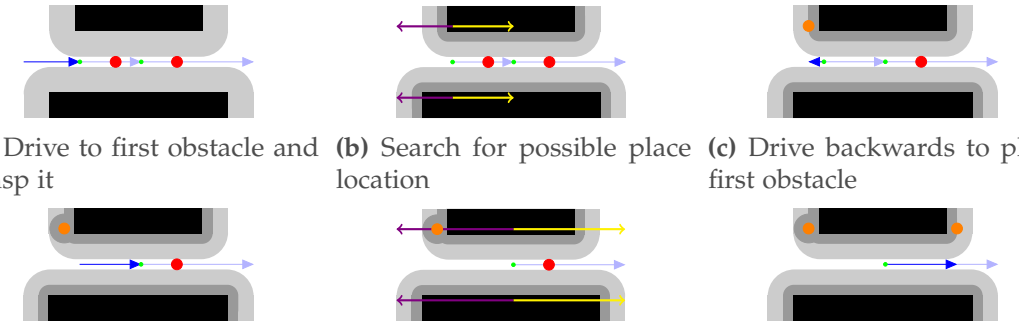


Figure D.5: The movable obstacle (red) can be grasped by the ground robot from positions on a circle around it. To avoid a collision with the fixed obstacle (black), the robot center cannot move inside the inflated area (gray). Thus, only the grasp locations marked in green are feasible.

Algorithm 2 Mission Planning

```
calculate lower bound on cost to go from each obstacle
min_cost ← ∞
add empty mission with start pose to Priority Queue (PQ)
while cost(PQ.top()) < min_cost do
  current_mission ← PQ.top()
  add remaining obstacles to map
  for all paths to these obstacles do
    if lower bound of new mission < min_cost then
      new_mission ← current_mission + path
      add new_mission to PQ
  compute path to goal
  if path is feasible then
    current_mission ← current_mission + path
    refine mission
    if cost(current_mission) < min_cost then
      min_cost ← mission_cost
      best_mission ← current_mission
```



(a) Drive to first obstacle and grasp it

(b) Search for possible place location

(c) Drive backwards to place first obstacle

(d) Drive to second obstacle and grasp it

(e) Search for possible place location

(f) Drive and place second obstacle

Figure D.6: To illustrate our obstacle-placing algorithm, we consider a narrow corridor with two movable obstacles inside. Fixed obstacles (black) are inflated to mark the area where the robot cannot drive (light gray) or cannot place obstacles (dark gray). Green dots indicate the grasp locations of movable obstacles (red). Since we place obstacles to the side of the robot, we search along parallel lines of the path (blue) for place locations. First, we search in forward direction (yellow) up to the next grasp location. If we cannot find a place location there, we start searching in backward direction (violet). The chosen obstacle place position is indicated in orange. The time for driving backwards is taken into account by our mission-planning algorithm which possibly affects the optimal mission (cf. Figures D.9a and D.9b). An example of a corridor with more movable obstacles is shown in Figure D.10.

D.5.1 Place Positions of Removed Obstacles

When removing an obstacle, we ideally try to place it on the side of the ground robot while the robot stands still in order to save time. Nonetheless, in narrow passages (e.g., in a passage as in Figure D.6), the robot cannot just place the obstacle to the left or right, but needs to carry it to a feasible place location. To do so, we search for unoccupied place locations along two lines parallel to the driving path. First, we search in forward direction up to the next obstacle grasp location. If we cannot find a feasible place position in that direction, we then also search in backward direction. This happens for example if two obstacles are in a narrow passage (cf. Figure D.6): the first obstacle must be carried out of the passage by driving backwards. The second obstacle can be placed when exiting the passage.

D.5.2 Path Refinement

We search paths on the grid map using single-source Dijkstra's algorithm. However, these paths should be smoothed for a ground robot to be executed (see Figure D.7). We therefore refine these paths using a simple algorithm (see Algorithm 3).

Although there exist more sophisticated planning algorithms on grid maps (such as Theta* [122]), this is of minor concern in our situation, where the robot is much larger than the grid resolution.

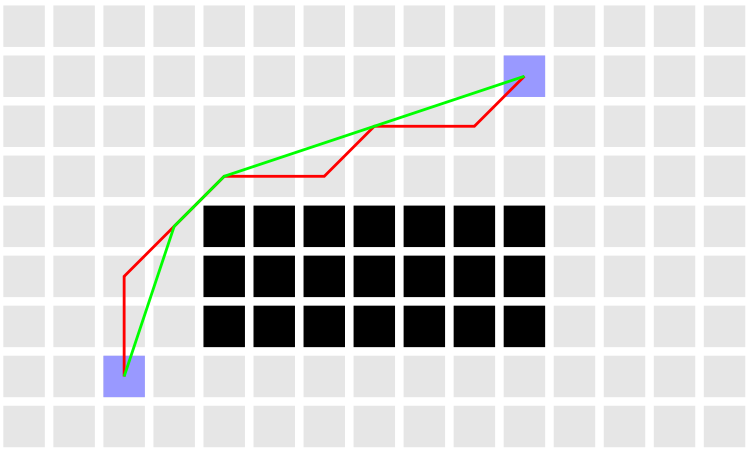


Figure D.7: The output of Dijkstra's algorithm (red) is refined (green) to find a smoother trajectory for the ground robot.

Algorithm 3 Optimizing Grid Map Paths

```
add first cell of old path to new path
for all cells in old path do
  if no line of sight to last cell in new path then
    add previous cell of old path to new path
add last cell of old path to new path
for all cells in new path do
  update orientation such that it points to next cell
```

D.6 Mission Execution

A feasible mission plan for the ground robot consists of a series of actions such as driving straight line path segments, removing obstacles, and delivering the first-aid kit. Each of these actions for the ground robot are commanded by the aerial robot in an iterative fashion. Once an action is commanded, the ground robot executes it without any external feedback. The aerial robot is then following the ground robot to command the next action. Commanding the ground robot in this iterative fashion to perform open-loop maneuvers eliminates problems with communication delays. Furthermore, it keeps the required communication at a minimum.

When the ground robot has to drive a straight line path segment, it is first localized in the map by the aerial robot. It is then commanded to execute a motion relative to the current location using its wheel odometry without feedback from the aerial robot. When the ground robot has executed the open-loop motion, it is again localized in the map by the aerial robot to compensate for accumulated drift of the wheel odometry. To remove an obstacle, the ground robot is commanded to drive to the chosen grasp location such that it can reach it with its gripper. When in front of the obstacle, the ground robot makes use of its laser scanner to measure the relative position of the obstacle precisely. It then grasps the obstacle and places it to a location that is again commanded by the aerial robot. After removing all the obstacles in the way, the ground robot can approach the goal location. Since we want to deliver the first-aid kit directly onto the goal location, the ground robot stops in front of the goal location such that it is within the gripper's reach (cf. Figure D.8). Once it is there, the ground robot is commanded to place the first-aid kit.

D.7 Results

In this section, we evaluate both our mission planner and the overall system performance. We analyze the mission plans for special, engineered cases. Further, we evaluate the computation time depending on the number of movable obstacles in the scene. Finally, we present the overall system performance during many demonstrations at a trade fair.

As parameters for the mission planning, we used $0.2\,\mathrm{m\,s^{-1}}$ driving speed, $30\,^{\circ}\,\mathrm{s^{-1}}$ rotational speed, and 15 s for grasping and placing an obstacle. We set the driving speeds of the youBot accordingly and measured the required grasp-and-place. For creating the grid map, we used a cell size of 5 cm.

D.7.1 Mission Planning

We evaluate our mission-planning algorithm using special and random cases. The visualization of the output is explained in Figure D.8.

We demonstrate the influence on the optimal mission when driving backwards to place an obstacle in Figure D.9. Even if the south path in Figure D.9a is shorter, the mission time would be longer due to the time required for additional driving backwards to place the first obstacle. However, if the north path is blocked, the mission planner will choose the other path (see Figure D.9b). If many obstacles are in a small corridor as in Figure D.10, the ground robot must carry all but one out of it backwards to be able to traverse the corridor.

In Figure D.11, we demonstrate the influence of the time required to remove an obstacle on the optimal mission.

Randomly generated scenes and the respective missions are shown in Figure D.12.

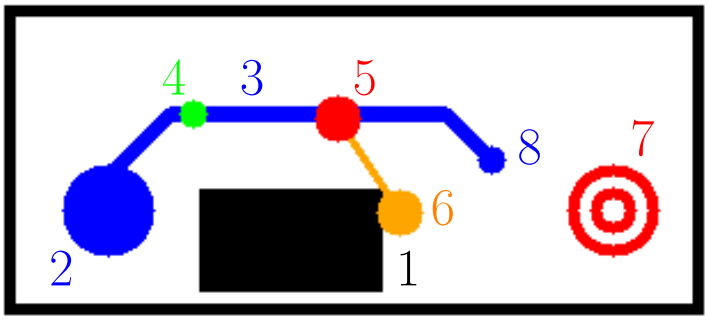
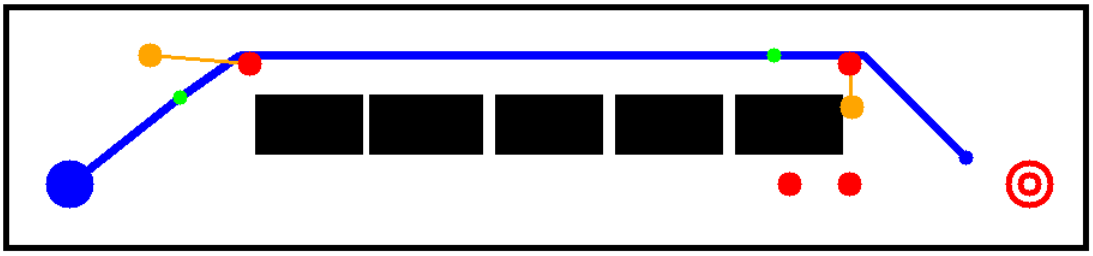


Figure D.8: Explanation of mission planner output: fixed obstacle (1), start location of the ground robot (2), path to drive (3), grasp location (4) of movable obstacle (5), place location (6), and goal (7). From the end of the path (8), the first-aid kit is delivered to the goal.

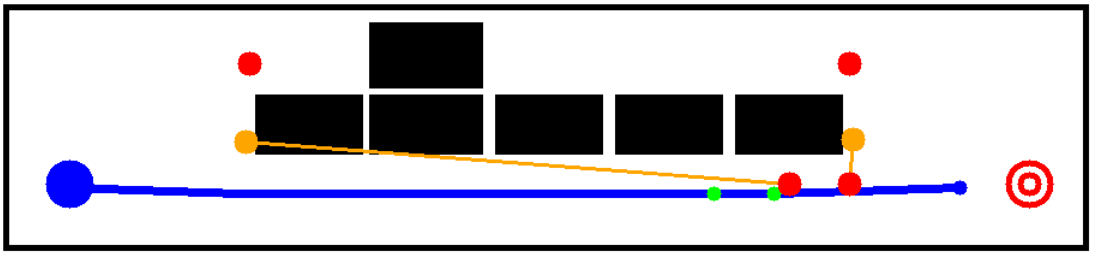
To evaluate the computational performance of our mission planner, we fixed a scene (cf. Figure D.12a) and varied the number of movable obstacles. For every chosen number of movable obstacles we run 50 trials with randomly placed movable obstacles to measure the computation time. The results are shown in Figure D.13. In two out of 250 cases, the planner could not find a feasible mission.

D.7.2 Overall System Performance

Our system was demonstrated at the AUTOMATICA'14 trade fair in Munich on four subsequent days, once every hour (see Figure D.14). The entire setup was dismounted after every demonstration. Both the movable and fixed obstacles were placed randomly before each run. The mission was accomplished successfully 23 out of 27 times (81%).



(a) Our mission-planning algorithm computes the fastest mission and not the shortest path. Therefore, it chooses the north path since it requires no driving backwards to place obstacles.



(b) If the path of Figure D.9a is blocked by an additional fixed obstacle, the south path is chosen. Note that this mission takes longer than the one above, since the robot has to drive backwards to place the obstacle.

Figure D.9: Placing movable obstacle might require additional backwards driving and, thus, increase the mission time. Therefore, the output of our mission-planning algorithm can be different from the shortest path. Color-coding is explained in Figure D.8.

Reasons for failures were mapping errors and wireless communication issues. In all these failure cases, our system reacted in a fail-safe way: the robots stopped to move and the error was reported to the operator.

D.8 Conclusion

In this paper, we demonstrated the autonomous collaboration of an aerial and a ground robot in a mock-up disaster scenario. We detailed the algorithm for mission planning for the ground robot and evaluated it for both special and random scenarios. The high success rate during a trade fair showed the robustness of our system.

Wireless communication is a major concern when dealing with multiple robots. We tackled this by performing all crucial computations onboard the robots. Thus, we only need to communicate high-level commands and sparse information, which do not require low-latency or high-bandwidth communication links.

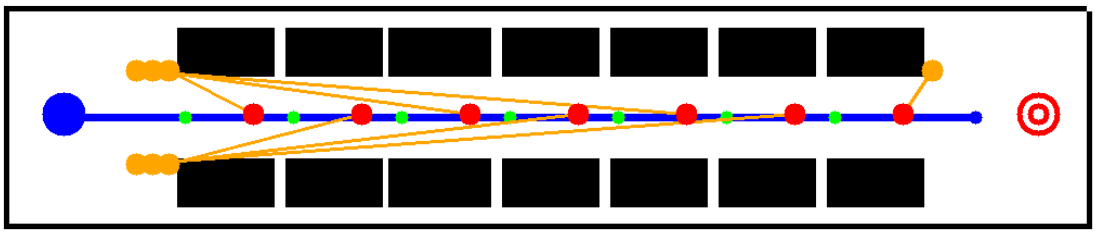
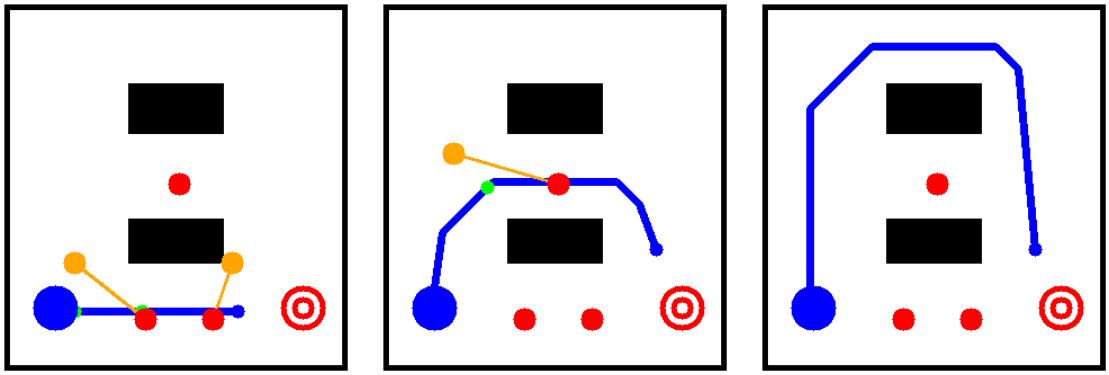


Figure D.10: All but one obstacle in a narrow corridor must be carried back to where the robot entered the corridor. Only the last obstacle is placed at the end of the corridor. Color-coding is explained in Figure D.8.



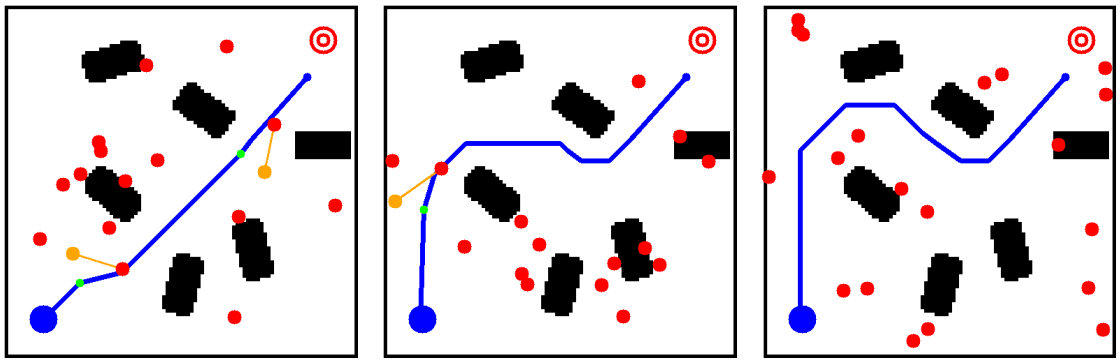
(a) Removal time 5 s **(b)** Removal time 11 s **(c)** Removal time 15 s

Figure D.11: The time-optimal mission depends on the time required to remove an obstacle. Color-coding is explained in Figure D.8.

D.8.1 Future Work

In real-world scenarios, two main assumptions of this paper are not valid: first, in our case, obstacles are marked with tags. Second, we assume the world to be flat and, therefore, plan missions only in 2D. We plan to overcome these limitations by building a 3D map using the images from the aerial robot, e.g., using our real-time dense reconstruction algorithm (REMODE) [133]. The maps from the aerial robot could also be combined with the ones from the ground robot [46]. Since wheeled robots, such as the KUKA youBot used here, are limited to flat surfaces, we aim at deploying legged robots for such missions.

Other directions of research include covering larger areas using fixed-wing aerial robots, extending the setup to multiple aerial and ground robots, computing and executing mission plans already while the flying robot is mapping, and adapting the mission plan when the environment changes during execution.



(a) 15 movable obstacles **(b)** 15 movable obstacles **(c)** 20 movable obstacles

Figure D.12: Movable obstacles are placed randomly in a scene to evaluate the computation time of our mission-planning algorithm. Results are shown in Figure D.13 and color-coding is explained in Figure D.8.

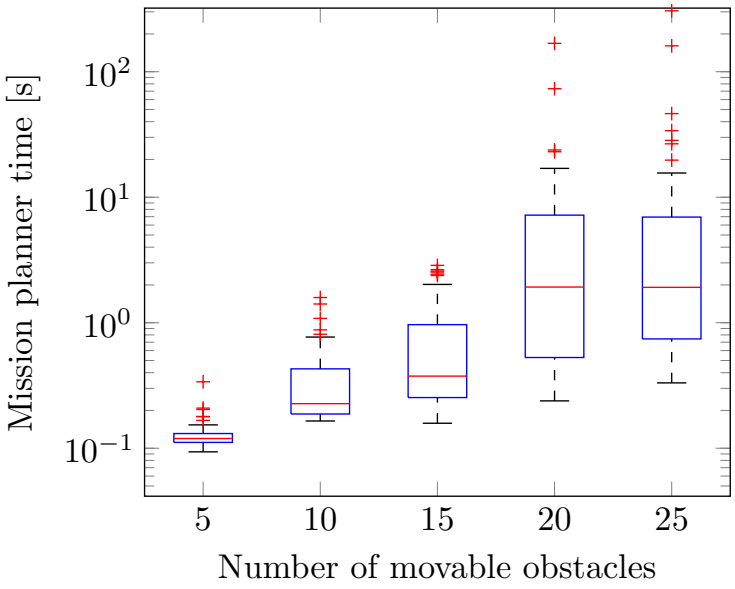


Figure D.13: Computation time of our mission planning algorithm for different numbers of movable obstacles. For every chosen number of movable obstacles we run 50 trials while placing the obstacles randomly. Example scenes are shown in Figure D.12.

Figure D.14: Demonstration of our system at the AUTOMATICA trade fair in Munich, Germany in June 2014.

E Aggressive Flight through Narrow Gaps

D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017. DOI: 10.1109/icra.2017.7989679

Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing using Active Vision

Davide Falanga, Elias Mueggler, Matthias Faessler and Davide Scaramuzza

Abstract — We address one of the main challenges towards autonomous quadrotor flight in complex environments, which is flight through narrow gaps. While previous works relied on off-board localization systems or on accurate prior knowledge of the gap position and orientation in the world reference frame, we rely solely on onboard sensing and computing and estimate the full state by fusing gap detection from a single onboard camera with an IMU. This problem is challenging for two reasons: (i) the quadrotor pose uncertainty with respect to the gap increases quadratically with the distance from the gap; (ii) the quadrotor has to actively control its orientation towards the gap to enable state estimation (i.e., active vision). We solve this problem by generating a trajectory that considers geometric, dynamic, and perception constraints: during the approach maneuver, the quadrotor always faces the gap to allow state estimation, while respecting the vehicle dynamics; during the traverse through the gap, the distance of the quadrotor to the edges of the gap is maximized. Furthermore, we replan the trajectory during its execution to cope with the varying uncertainty of the state estimate. We successfully evaluate and demonstrate the proposed approach in many real experiments, achieving a success rate of 80% and gap orientations up to 45°. To the best of our knowledge, this is the first work that addresses and achieves autonomous, aggressive flight through narrow gaps using only onboard sensing and computing and without prior knowledge of the pose of the gap.

Supplementary Material

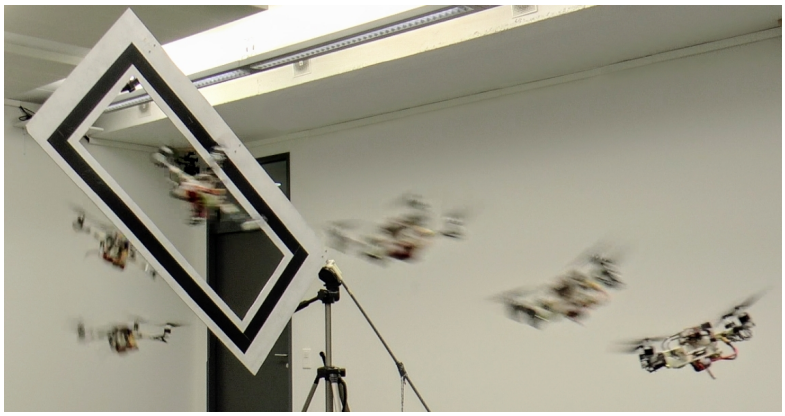
The accompanying video is available at:
http://rpg.ifi.uzh.ch/aggressive_flight.html

E.1 Introduction

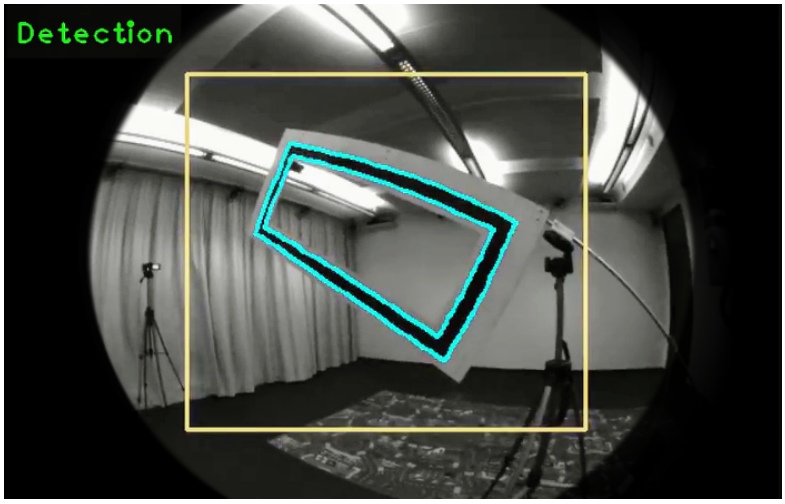
Recent works have demonstrated that micro quadrotors are extremely agile and versatile vehicles, able to execute very complex maneuvers [115, 31, 107]. These demonstrations highlight that one day quadrotors could be used in search and rescue applications, such as in the aftermath of an earthquake, to navigate through buildings, by entering and exiting through narrow gaps, and to quickly localize victims.

In this paper, we address one of the main challenges towards autonomous quadrotor flight in complex environments, which is flight through narrow gaps. What makes this problem challenging is that the gap is very small, such that precise trajectory-following is required, and can be oriented arbitrarily, such that the quadrotor cannot fly through it in near-hover conditions. This makes it necessary to execute an aggressive trajectory (i.e., with high velocity and angular accelerations) in order to align the vehicle to the gap orientation (cf. Fig. E.1).

Previous works on aggressive flight through narrow gaps have focused solely on the control and planning problem and therefore relied on accurate state estimation from external motion-capture systems and/or accurate knowledge of the gap position and orientation in the world reference frame. Since these systems were not gap-aware, the trajectory was generated before execution and never replanned. Therefore, errors in the measure of the pose of the gap in the world frame were not taken into account, which may lead to a collision with gap. Conversely, we are interested in using only *onboard sensing and computing*, *without any prior knowledge* of the gap pose in the world frame. More specifically, we address the case where state estimation is done by fusing gap detection through a single, forward-facing camera with an IMU. We show that this raises an interesting *active-vision* problem (i.e, coupled perception and control). Indeed, for the robot to localize with respect to the gap, a trajectory that guarantees that the quadrotor always faces the gap must be selected (perception constraint). Additionally, it must be replanned multiple times during its execution to cope with the varying uncertainty of the state estimate, which is quadratic with the distance from the gap. Furthermore, during the traverse, the quadrotor must maximize the distance from the edges of the gap (geometric constraint) to avoid collisions. At the same time, it must do so without relying on any visual feedback (when the robot is very close to the gap, it exits from the field of view of the camera). Finally, the trajectory must be feasible with respect to the dynamic constraints of the vehicle.



(a) The quadrotor passing through the gap.



(b) View from the onboard camera

Figure E.1: Sequence of our quadrotor passing through a narrow, 45°-inclined gap. Our state estimation fuses gap detection from a single onboard forward-facing camera with an IMU. All planning, sensing, control run fully onboard on a smartphone computer.

Our proposed trajectory generation approach is independent of the gap-detection algorithm being used; thus, to simplify the perception task, we use a gap with a black-and-white rectangular pattern (cf. Fig. E.1) for evaluation and demonstration.

E.1.1 Related Work

A solution for trajectory planning and control for aggressive quadrotor flight was presented in [107]. The authors demonstrated their results with aggressive flight through a narrow gap, and by perching on inclined surfaces. The quadrotor state was obtained using a motion-capture system. To fly through a narrow gap, the vehicle started by hovering in a pre-computed position, flew a straight line towards a launch point, and then controlled its orientation to align with the gap. The method was not

plug-and-play since it needed training through *iterative learning* in order to refine the launch position and velocity. This was due to the instantaneous changes in velocity caused by the choice of a straight line for the approach trajectory. Unlike their method, we use a technique that computes polynomial trajectories which are guaranteed to be feasible with respect to the control inputs. The result is a smooth trajectory, compatible with the quadrotor dynamic constraints, which makes learning unnecessary. Indeed, in realistic scenarios, such as search-and-rescue missions, we cannot afford training but must pass on the first attempt.

In [106], the same authors introduced a method to compute trajectories for a quadrotor solving a Quadratic Program, which minimizes the snap (i.e., the fourth derivative of position). In their experiments, agile maneuvers, such as passing through a hula-hoop thrown by hand in the air, were demonstrated using state estimation from a motion-capture system.

In [164], a technique that lets a quadrotor pass through a narrow gap while carrying a cable-suspended payload was presented and was experimentally validated using a motion-capture system for state estimation.

In [124], the authors proposed an unconstrained nonlinear model predictive control algorithm in which trajectory generation and tracking are treated as a single, unified problem. The proposed method was validated in a number of experiments, including a rotorcraft passing through an inclined gap. Like the previous systems, they used a motion-capture system for state estimation.

In [99], the authors proposed a vision-based method for autonomous flight through narrow gaps by fusing data from a downward and a forward-looking camera, and an IMU. Trajectory planning was executed on an external computer. However, the authors only considered the case of an horizontal gap, therefore no agile maneuver was necessary.

In [94], the authors proposed methods for onboard vision-based state estimation, planning, and control for small quadrotors, and validated the approach in a number of agile maneuvers, among which flying through an inclined gap. Since state estimation was performed by fusing input from a downward-looking camera and an IMU, rather than from gap detection, the gap position and orientation in the world reference frame had to be measured very accurately prior to the execution of the maneuver. The trajectory was generated before execution and never replanned. Therefore, errors in the measure of the pose of the gap in the world frame were not taken into account, which may lead to a collision with gap. To deal with this issue, the authors used a gap considerably larger than the vehicle size.

All the related works previously mentioned relied on the accurate state estimates from a motion-capture system or accurate prior knowledge of the gap position and

orientation in the world reference frame. Additionally, in all these works but [124] and [94] trajectory generation was performed on an external computer. The advantages of a motion-capture system over onboard vision are that the state estimate is always available, at high frequency, accurate to the millimeter, and with almost *constant noise covariance* within the tracking volume. Conversely, a state estimate from onboard vision can be intermittent (e.g., due to misdetections); furthermore, its covariance increases *quadratically* with the distance from the scene and is strongly affected by the type of structure and texture of the scene. Therefore, to execute a complex aggressive maneuver, like the one tackled in this paper, while using only onboard sensing and *gap-aware* state estimation, it becomes necessary *to couple perception with the trajectory generation process* (i.e., active vision). Specifically, the desired trajectory has to render the gap always visible by the onboard camera in order to estimate its relative pose.

E.1.2 Contributions

Our method differs from previous works in the following aspects: (i) we rely solely on onboard, visual-inertial sensors and computing, (ii) we generate a trajectory that facilitates the perception task, while satisfying geometric and dynamic constraints, and (iii) we do not require iterative learning, neither do we need to know a priori the gap position and orientation in the world frame. To the best of our knowledge, this is the first work that addresses and achieves aggressive flight through narrow gaps with state estimation via gap detection from an onboard camera and IMU.

The remainder of this paper is organized as follows. Section E.2 presents the proposed trajectory-generation algorithm. Section E.3 describes the state-estimation pipeline. Section E.4 presents the experimental results. Section E.5 discusses the results and provides additional insights about the approach. Finally, Section E.6 draws the conclusions.

E.2 Trajectory Planning

We split the trajectory planning into two consecutive stages. First, we compute a traverse trajectory to pass through the gap. This trajectory maximizes the distance from the vehicle to the edges of the gap in order to minimize the risk of collision. In a second stage, we compute an approach trajectory in order to fly the quadrotor from its current hovering position to the desired state that is required to initiate the traverse trajectory. While both trajectories need to satisfy *dynamic constraints*, the approach trajectory also satisfies *perception constraints*, i.e., it lets the vehicle-mounted camera always face the gap. This is necessary to enable state estimation with respect to the gap.

E.2.1 Traverse Trajectory

During the gap traversal, the quadrotor has to minimize the risk of collision. We achieve this by forcing the traverse trajectory to intersect the center of the gap while simultaneously lying in a plane orthogonal to the gap (see Fig. E.2). In the following, we derive the traverse trajectory in this orthogonal plane and then transform it to the 3D space.

Let W be our world frame. The vector $\mathbf{p}_G$ and the rotation matrix $\mathbf{R}_G$ denote the position of the geometric center of the gap and its orientation with respect to W, respectively. Let Π be a plane orthogonal to the gap, passing through its center and parallel to the longest side of the gap (cf. Fig. E.2). Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be the unit vectors spanning such a plane Π, whose normal unit vector is $\mathbf{e}_3$. The $\mathbf{e}_2$ axis is orthogonal to the gap and $\mathbf{e}_1 = \mathbf{e}_2 \times \mathbf{e}_3$.

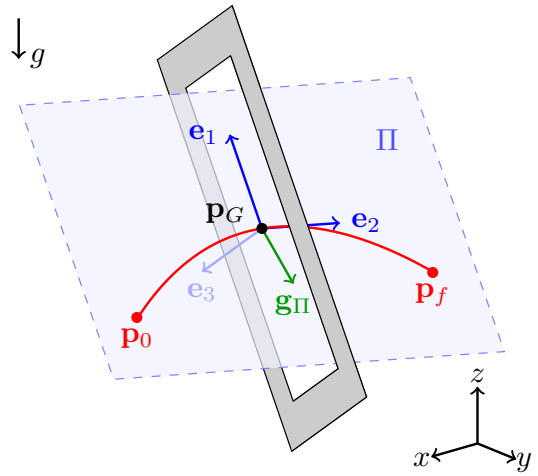


Figure E.2: An inclined gap and the corresponding plane Π.

Intuitively, a trajectory that lies in the plane Π and passes through the center of the gap, minimizes the risk of impact with the gap.

To constrain the motion of the vehicle to the plane Π, it is necessary to compensate the projection of the gravity vector $\mathbf{g}$ onto its normal vector $\mathbf{e}_3$. Therefore, a constant thrust of magnitude $\langle \mathbf{g}, \mathbf{e}_3 \rangle$ needs to be applied orthogonally to Π. By doing this, a 2D description of the quadrotor's motion in this plane is sufficient. The remaining components of $\mathbf{g}$ in the plane Π are computed as

$$\mathbf{g}_\Pi = \mathbf{g} - \langle \mathbf{g}, \mathbf{e}_3 \rangle \, \mathbf{e}_3. \tag{E.1}$$

Since this is a constant acceleration, the motion of the vehicle along Π is described by

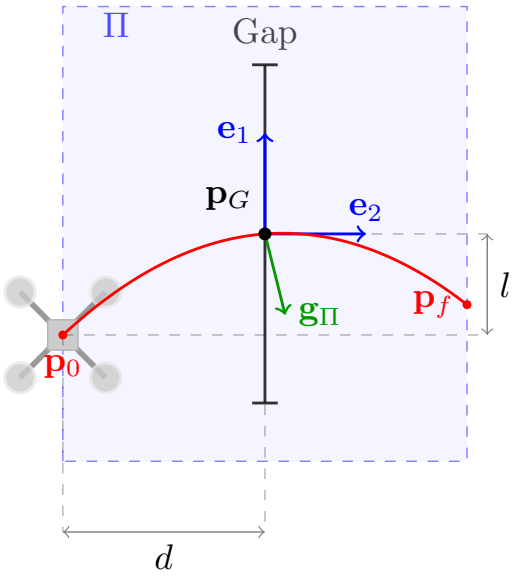


Figure E.3: The traverse trajectory in the plane Π.

the following second order polynomial equation:

$$p_i(t) = p_i(t_0) + v_i(t_0)t + \frac{1}{2}g_{\Pi,i}t^2, \tag{E.2a}$$

$$v_i(t) = v_i(t_0) + g_{\Pi,i}t, \tag{E.2b}$$

where the subscript $i = \{1, 2\}$ indicates the component along the $\mathbf{e}_i$ axis. The quadrotor enters the traverse trajectory at time t_0, t is the current time, and p and v denote its position and velocity, respectively.

Equation (E.2) describes a ballistic trajectory. When $g_{\Pi,2} = 0$, it is the composition of a uniformly accelerated and a uniform-velocity motion. In other words, in these cases the quadrotor moves on a parabola in space.

Let l and d be the distance between $\mathbf{p}_G$ and the initial point of the trajectory, $\mathbf{p}_0$, along $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively (cf. Fig. E.3). These two parameters determine the initial position and velocity in the plane Π, as well as the time t_c necessary to reach $\mathbf{p}_G$. The values of d and l are determined through an optimization problem, as explained later in Sec. E.2.2.

For a generic orientation $\mathbf{R}_G$ of the gap, (E.2) is characterized by a uniformly accelerated motion along both the axes $\mathbf{e}_1$ and $\mathbf{e}_2$. Therefore, it is not possible to guarantee that the distance traveled along the $\mathbf{e}_2$ axis before and after the center of the gap are equal while also guaranteeing that the initial and final position have the same coordinate along the $\mathbf{e}_1$ axis. For safety reasons, we prefer to constrain the motion along the $\mathbf{e}_2$ axes, i.e., orthogonally to the gap, such that the distances traveled before and after the

gap are equal.

Given the components of the unit vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ in the world frame, it is now possible to compute the initial conditions $\mathbf{p}_0 = \mathbf{p}(t_0)$ and $\mathbf{v}_0 = \mathbf{v}(t_0)$ in 3D space as follows:

$$\mathbf{p}_0 = \mathbf{p}_G - l\mathbf{e}_1 - d\mathbf{e}_2, \tag{E.3a}$$

$$\mathbf{v}_0 = \left(\frac{l}{t_c} - \frac{1}{2}g_{\Pi,1}t_c\right)\mathbf{e}_1 + \left(\frac{d}{t_c} - \frac{1}{2}g_{\Pi,2}t_c\right)\mathbf{e}_2, \tag{E.3b}$$

where:

$$t_c = \sqrt{\frac{-2l}{g_{\Pi,1}}} \tag{E.4}$$

is the time necessary to reach the center of the gap once the traverse trajectory starts.

Note that this solution holds if $g_{\Pi,2} \geq 0$ which applies if $\mathbf{e}_2$ is horizontal or pointing downwards in world coordinates. The case $g_{\Pi,2} < 0$ leads to similar equations, which we omit for brevity. The final three-dimensional trajectory then has the following form:

$$\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}_\Pi t^2, \tag{E.5a}$$

$$\mathbf{v}(t) = \mathbf{v}_0 + \mathbf{g}_\Pi t, \tag{E.5b}$$

$$\mathbf{a}(t) = \mathbf{g}_\Pi. \tag{E.5c}$$

This trajectory is inexpensive to compute since it is solved in closed form. Also, note that during the traverse the gap is no longer detectable. Nevertheless, since the traverse trajectory is short and only requires *constant control inputs* (a thrust of magnitude $\langle \mathbf{g}, \mathbf{e}_3 \rangle$ and zero angular velocities), it is possible to track it accurately enough to not collide with the gap, even without any visual feedback.

E.2.2 Optimization of the Traverse Trajectory

To safely pass through the gap, the quadrotor must reach the initial position and velocity of the traverse trajectory described by (E.3a)-(E.3b) with an acceleration equal to $\mathbf{g}_\Pi$ at time t_0. An error in these initial conditions is propagated through time according to (E.5a)-(E.5c), and therefore may lead to a collision. The only viable way to reduce the risk of impact is to reduce the time duration of the traverse. More specifically, (E.4) shows that one can optimize the value of l to reduce the time of flight of the traverse trajectory. On the other hand, (E.3b) and (E.4) show that reducing l leads to an increase in the norm of the initial velocity $\mathbf{v}_0$. Intuitively speaking, this is due to the fact that,

for a given value of d, if the time of flight decreases, the velocity along the $\mathbf{e}_2$ axis has to increase to let the vehicle cover the same distance in a shorter time. The initial velocity also depends on d, which can be tuned to reduce the velocity at the start of the traverse. The value of d cannot be chosen arbitrarily small for two reasons: (i) it is necessary to guarantee a safety margin between the quadrotor and the gap at the beginning of the traverse; (ii) the gap might not be visible during the final part of the approach trajectory. For this reason, we compute the values of the traverse trajectory parameters solving the following optimization problem:

$$\min_{d,l} t_c \quad \text{s.t.} \quad \|\mathbf{v}_0\| \leq \mathrm{v}_{0,\max}, \quad d \geq d_{\min}, \tag{E.6}$$

where $\mathrm{v}_{0,\max}$ and $d_{\min}$ are the maximum velocity allowed at the start of the traverse and the minimum value of d, respectively. We solve the nonlinear optimization problem described by (E.6) with Sequential Quadratic Programming (SQP [84], using to the NLopt library [74]. Thanks to the small dimensionality of the problem, it can be solved onboard in few tens of milliseconds.

E.2.3 Approach Trajectory

Once the traverse trajectory has been computed, its initial conditions (namely, position, velocity, and acceleration) are known. Now we can compute an approach trajectory from a suitable start position to these initial conditions. Note that this start position is not the current hover position but also results from the proposed trajectory generation method. Our goal in this step is to find a trajectory that not only matches the initial conditions of the traverse trajectory, but also enables robust perception and state estimation with respect to the gap.

Robust state estimation with respect to the gap can only be achieved by always keeping the gap in the field of view of a forward-facing camera onboard the quadrotor. Since it is difficult to incorporate these constraints into the trajectory generation directly, we first compute trajectory candidates and then evaluate their suitability for the given perception task. To do so, we use the approach proposed in [118], where a fast method to generate feasible trajectories for flying robots is presented. In that paper, the authors provide both a closed-form solution for motion primitives that minimize the jerk and a feasibility check on the collective thrust and angular velocities. The benefit of using such a method is twofold. First, it allows us to obtain a wide variety of candidate trajectories within a very short amount of time by uniformly sampling the start position and the execution time within suitable ranges. This way we can quickly evaluate a large set of candidate trajectories and select the best one according to the optimality criterion described in Sec. E.2.5. Each of these candidate trajectories consists of the quadrotor's 3D position and its derivatives. Second, and most importantly, since the computation and the verification of each trajectory takes on average a two tenths of millisecond, it is

possible to replan the approach trajectory at each control step, counteracting the effects of the uncertainty in the pose estimation of the quadrotor when it is far away from the gap. Each new approach trajectory is computed using the last state estimate available. In the following, we describe how we plan a yaw-angle trajectory for each candidate and how we select the best candidate to be executed.

E.2.4 Yaw-Angle Planning

In [106], the authors proved that the dynamic model of a quadrotor is *differentially flat*. Among other things, this means that the yaw angle of the quadrotor can be controlled independently of the position and its derivatives. In this section, we present how to compute the yaw angle such that a camera mounted on the quadrotor always faces the gap. Ideally, the camera should be oriented such that the center of the gap is projected as close as possible to the center of the image, which yields the maximum robustness for visual state estimation with respect to the gap against disturbances on the quadrotor.

To compute the desired yaw angle, we first need to compute the ideal orientation of the camera. Let $\mathbf{p}_G$ be the coordinates of the center of the gap with respect to the world frame W. Furthermore, let $\mathbf{R}_{WC}$ and $\mathbf{p}_C$ be the extrinsic parameters of the camera: $\mathbf{p}_C$ is the camera's position and the rotation matrix $\mathbf{R}_{WC} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ defines the camera orientation with respect to the world frame, where $\mathbf{r}_3$ is the camera's optical axis.

For a given trajectory point, we can compute the vector from the camera to the center of the gap $\mathbf{d} = \mathbf{p}_G - \mathbf{p}_C$. Ideally, we can now align the camera's optical axis $\mathbf{r}_3$ with $\mathbf{d}$ but since the trajectory constrains the quadrotor's vertical axis $\mathbf{z}_b$, we can generally not do this. Therefore, we minimize the angle between $\mathbf{d}$ and $\mathbf{r}_3$ by solving the following constrained optimization problem:

$$\mathbf{r}_3^* = \arg\max_{\mathbf{x}} \langle \mathbf{x}, \mathbf{d} \rangle \quad \text{s.t. } \|\mathbf{x}\| = 1, \quad \langle \mathbf{x}, \mathbf{z}_b \rangle = k, \tag{E.7}$$

where the last constraint says that the angle between the quadrotor's vertical body axis $\mathbf{z}_b$ and the camera's optical axis is constant and depends on how the camera is mounted on the vehicle. For example, $k = 0$ if the camera is orthogonal to the $\mathbf{z}_b$ axis as it is the case in our setup with a forward-facing camera.

Letting $\mathbf{d}_{\perp z_b} = \mathbf{d} - \langle \mathbf{d}, \mathbf{z}_b \rangle \mathbf{z}_b$ be the component of $\mathbf{d}$ perpendicular to $\mathbf{z}_b$, the solution of (E.7) is

$$\mathbf{r}_3^* = \sqrt{1 - k^2} \frac{\mathbf{d}_{\perp z_b}}{\|\mathbf{d}_{\perp z_b}\|} + k\mathbf{z}_b, \tag{E.8}$$

which is a vector lying in the plane spanned by $\mathbf{d}$ and $\mathbf{z}_b$, and the minimum angle

between the ideal and the desired optical axis is $\arccos(\langle \mathbf{r}_3^*, \mathbf{d} \rangle / \|\mathbf{d}\|)$, i.e.,

$$\theta_{min} = \arccos\Big((\sqrt{1-k^2}\|\mathbf{d}_{\perp z_b}\| + k \langle \mathbf{d}, \mathbf{z}_b \rangle) / \|\mathbf{d}\| \Big). \tag{E.9}$$

Once $\mathbf{r}_3^*$ is known, we can compute the yaw angle such that the actual camera optical axis $\mathbf{r}_3$ is aligned with $\mathbf{r}_3^*$.

Observe that in the particular case of a trajectory point that allows to align $\mathbf{r}_3$ with $\mathbf{d}$, we have $\langle \mathbf{d}, \mathbf{z}_b \rangle = k\|\mathbf{d}\|$ and the solution of (E.7) reduces to $\mathbf{r}_3^* = \frac{\mathbf{d}}{\|\mathbf{d}\|}$, with a minimum angle $\theta_{\text{min}} = \arccos(\langle \mathbf{r}_3, \mathbf{d} \rangle / \|\mathbf{d}\|) = \arccos(1) = 0$.

E.2.5 Selection of the Approach Trajectory to Execute

In the previous sections, we described how we compute a set of candidate trajectories in 3D space and yaw for approaching the gap. All the candidate trajectories differ in their start position and their execution time. From all the computed candidates, we select the one that provides the most reliable state estimate with respect to the gap. As a quality criterion for this, we define a cost function J composed of two terms:

- the Root Mean Square (RMS) θ_{rms} of (E.9) over every sample along a candidate trajectory;
- the straight-line distance d_0 to the gap at the start of the approach.

More specifically:

$$J = \frac{\theta_{\text{rms}}}{\bar{\theta}} + \frac{d_0}{\bar{d}}, \tag{E.10}$$

where $\bar{\theta}$ and $\bar{d}$ are normalization constants that make it possible to sum up quantities with different units, and render the cost function dimensionless. This way, the quadrotor executes the candidate approach trajectory that keeps the center of the gap as close as possible to the center of the image for the entire trajectory, and at the same time prevents the vehicle from starting too far away from the gap.

E.2.6 Recovery after the Gap

Since we localize the quadrotor with respect to the gap in order to traverse it, the quadrotor is left with no state estimate after the traversal. Therefore, at this point it has to recover a vision-based state estimate and then hover in a fixed position without colliding with the environment. We solve this problem using the automatic recovery system detailed in [37], where the authors provide a method to let a quadrotor stabilize automatically after an aggressive maneuver, e.g. after a manual throw in the air.

E.3 State Estimation

E.3.1 State Estimation from Gap Detection

Our proposed trajectory generation approach is independent of the gap-detection algorithm being used; thus, to simplify the perception task, we use a black-and-white rectangular pattern to detect the gap (cf. Fig. E.1). A valid alternative to cope with real-world gaps would be to use monocular dense-reconstruction methods, such as REMODE [133]; however, they require more computing power (GPUs).

We detect the gap in each image from the forward-facing camera by applying a sequence of steps: first, we run the Canny edge detector, undistort all edges, and group close edges [160]; then, we search for quadrangular shapes and run geometrical consistency checks. Namely, we search for a quadrangle that contains another one and check the area ratio of these two quadrangles. Finally, we refine the locations of the eight corners to sub-pixel accuracy using line intersection.

Since the metric size of the gap is known, we estimate the 6-DOF pose by solving a Perspective-n-Points (PnP) problem (where $n = 8$ in our case). As a verification step, we require that the reprojection error is small. We then refine the pose by minimizing also the reprojection error of all edge pixels. To speed up the computation, we only search the gap in a region of interest around the last detection. Only when no detection is found, the entire image is searched. The detector runs with a frequency of more than 30 Hz onboard the quadrotor.

Finally, we fuse the obtained pose with IMU measurements to provide a full state estimate using the multi-sensor fusion framework of [98].

E.4 Experiments

E.4.1 Experimental Setup

We tested the proposed framework on a custom-made quadrotor, assembled from off-the-shelf hardware, 3D printed parts, and self-designed electronic components (see Fig. E.4). The frame of the vehicle is composed of a 3D printed center cross and four carbon fiber profiles as arms. Actuation is guaranteed by four RCTimer MT2830 motors, controlled by Afro Slim ESC speed controllers. The motors are tilted by 15° to provide three times more yaw-control action, while only losing 3 % of the collective thrust.

Our quadrotor is equipped with a PX4FMU autopilot that contains an IMU and a micro controller on which our custom low-level controller runs. Trajectory planning, state estimation and high-level control run on an Odroid-XU4 single-board computer. Our algorithms have been implemented in ROS, running on Ubuntu 14.04. Communication

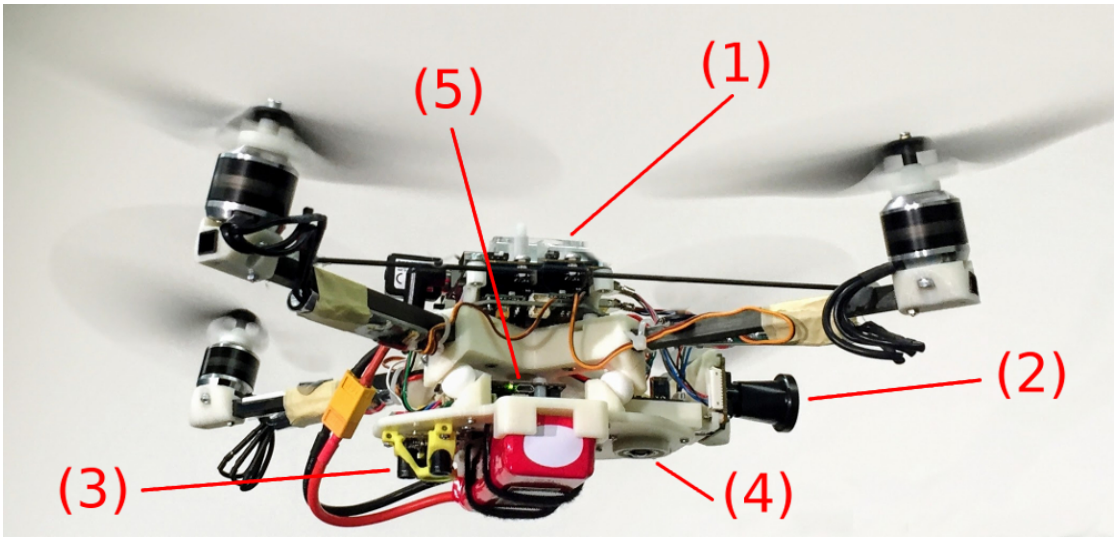


Figure E.4: The quadrotor platform used in the experiments. (1) Onboard computer. (2) Forward-facing fisheye camera. (3) TeraRanger One distance sensor and (4) downward-facing camera, both used *solely during the recovery phase*. (5) PX4 autopilot. The motors are tilted by 15° to provide three times more yaw-control action, while only losing 3 % of the collective thrust.

between the Odroid and the PX4 runs over *UART*.

Gap-detection is done through a forward-facing fisheye camera (MatrixVision mvBlueFOX-MLC200w 752 × 480-pixel monochrome camera with a 180° lens), which ensures that the gap can be tracked until very close. To allow the robot to execute the recovery maneuver after traversing the gap, we mounted the same hardware detailed in [37], which consists of a TeraRanger One distance sensor and a downward-facing camera. Notice, however, that these are *not used* for state estimation before passing the gap but *only* to recover and switch into stable hovering after the traverse.

The overall weight of the vehicle is 830 g, while its dimension are 55 × 12 cm (largest length measured between propeller tips). The dimensions of the rectangular gap are 80 × 28 cm. When the vehicle is at the center of the gap, the tolerances along the long side and short sides are only 12.5 cm, and 8 cm, respectively (cf. Fig. E.5). This highlights that the traverse trajectory must be followed with centimeter accuracy to avoid a collision.

The parameters of the traverse trajectory (Sec. E.2.2) have been set as $v_{0,max} = 3\,m\,s^{-1}$, $d_{min} = 0.25\,cm$. The normalization constants $\bar{\theta}$ and $\bar{d}$, introduced in Sec. E.2.5, have been manually tuned to let the quadrotor start the maneuver close enough to render vision-based pose estimation reliable and, at the same time, keep the gap as close as possible to the center of the image.

The dynamic model and the control algorithm used in this work are the same presented

Figure E.5: Our quadrotor during a traverse.

in [37]. We refer the reader to that for further details.

E.4.2 Results

To demonstrate the effectiveness of the proposed method, we flew our quadrotor through a gap inclined at different orientations. We consider both rotations around the world x and y axes, and denote them as *roll* and *pitch*, respectively. Overall, we ran 35 experiments with the roll angle ranging between 0° and 45° and the pitch angle between 0° and 30°. We discuss the choice of these values in Sec. E.5.3. With the gap inclined at 45°, the quadrotor reaches speeds of $3\,\mathrm{m\,s^{-1}}$ and angular velocities of $400\,^\circ\,\mathrm{s^{-1}}$.

We define an experiment as successful if the quadrotor passes through the gap without collision and recovers and locks to a hover position. We achieved a remarkable success rate of 80%. When failure occurred, we found this to be caused by a persistent absence of a pose estimate from the gap detector during the approach trajectory. This led to a large error in matching the initial conditions of the traverse trajectory, which resulted in a collision with the frame of the gap.

Figure E.6 shows the estimated position, velocity, and orientation against ground truth for some of the most significant experiments and for different orientations of the gap (namely: 20° roll, 0° pitch; 45° roll, 0° pitch; and 30° roll, 30° pitch). Ground truth is recorded from an OptiTrack motion-capture system. It can be observed that the desired trajectories were tracked remarkably well. Table E.1 reports the statistics of

	Position [m]			Velocity [m s^{-1}]			Orientation [°]	
	x	**y**	**z**	**x**	**y**	**z**	**roll**	**pitch**
μ	0.04	0.04	0.03	0.09	0.15	0.08	6.04	8.89
σ	0.03	0.02	0.03	0.08	0.10	0.06	3.70	5.85

Table E.1: Position, velocity and orientation error statistics at time $t = t_c$. The mean error μ and the standard deviation σ are computed using ground truth data gathered from 35 experiments conducted with the gap at different orientations.

the errors when the quadrotor passes through the plane in which the gap lies (i.e., at $t = t_c$), measured as the distance between actual and desired state. These statistics include both the successful and the unsuccessful experiments. The average of the norm of the position error at the center of the gap was 0.06 m, with a standard deviation of 0.05 m. The average of the norm of the velocity error was below 0.19 m s^{-1}, with a standard deviation of 0.20 m s^{-1}. We refer the reader to the attached video for further experiments with different orientations of the gap. Figure E.7 shows a picture of one of the experiments with the executed approach and traverse trajectories marked in color.

E.5 Discussion

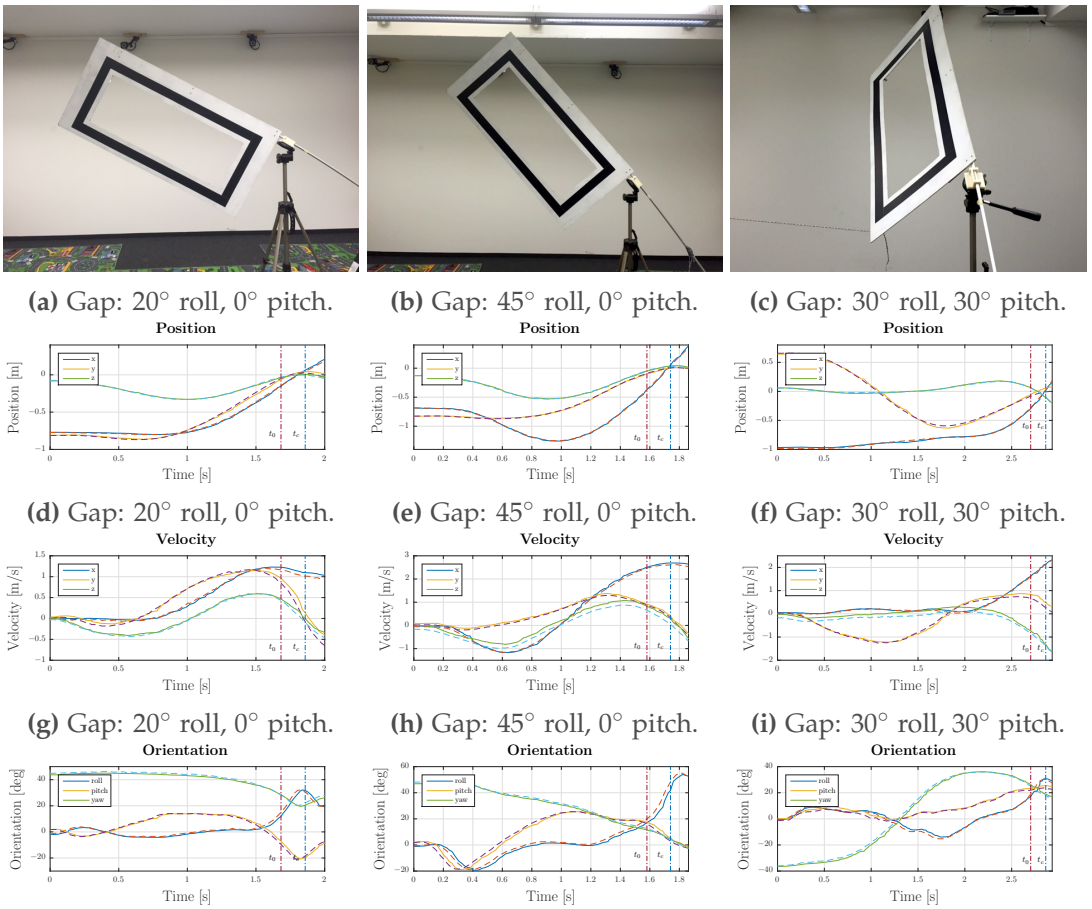
In this section, we discuss our approach and provide more insights into our experiments.

E.5.1 Replanning

The method we use to compute the approach meneuver [118] can fail to verify whether a trajectory is feasible or not, as also highlighted by the authors. This usually happens when the time duration of the trajectory is short. In such a case, we skip the replanning and provide the last available approach trajectory to our controller.

E.5.2 Trajectory Computation Times

The trajectory planning approach we adopt for the approach phase is fast enough to compute and test 40,000 trajectories in less then one second, even with the additional computational load induced by our check on the gap perception. The computation of each trajectory on the on-board computer takes on average (0.240 ± 0.106) ms, including: (i) generation of the trajectory; (ii) feasibility check; (iii) trajectory sampling and computation of the yaw angle for each sample; (iv) evaluation of the cost function described in (E.10); (v) comparison with the current best candidate. It is important to point out that these values do not apply to the *replanning* of the approach trajectory



(a) Gap: 20° roll, 0° pitch. **(b)** Gap: 45° roll, 0° pitch. **(c)** Gap: 30° roll, 30° pitch.

(d) Gap: 20° roll, 0° pitch. **(e)** Gap: 45° roll, 0° pitch. **(f)** Gap: 30° roll, 30° pitch.

(g) Gap: 20° roll, 0° pitch. **(h)** Gap: 45° roll, 0° pitch. **(i)** Gap: 30° roll, 30° pitch.

(j) Gap: 20° roll, 0° pitch. **(k)** Gap: 45° roll, 0° pitch. **(l)** Gap: 30° roll, 30° pitch.

Figure E.6: Comparison between ground truth and estimated position (top), velocity (center), and orientation (bottom). Each column depicts the result of an experiment conducted with a different configuration of the gap: d, g and j 20° of roll and 0° of pitch; e, h and k 45° of roll and 0° of pitch; f, i and l 30° of roll and 30° of pitch. The approach trajectory starts at $t = 0$ and ends at $t = t_0$, when the traverse trajectory is executed. The quadrotor reaches the center of the gap at $t = t_c$ and starts the recovery maneuver at the final time of each plot. We refer the reader to the accompanying video for further experiments with different orientations of the gap.

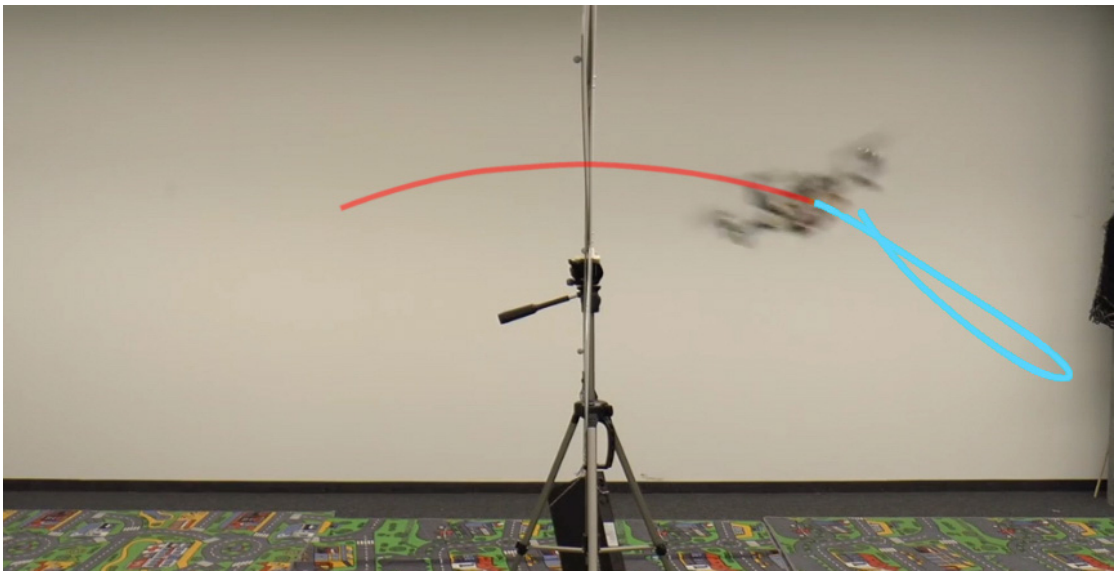


Figure E.7: Our quadrotor executing the whole trajectory split into approach (blue), traverse (red).

during its execution, since the initial state is constrained by the current state of the vehicle and there is no cost function to evaluate. In such a case, the computation is much faster and for each trajectory it only takes (0.018 ± 0.011) ms on average.

E.5.3 Gap configuration

Our trajectory generation formulation is able to provide feasible trajectories with any configuration of the gap, e.g., when the gap is perfectly vertical (90° roll angle) or perfectly horizontal (90° pitch angle). However, in our experiments we limit the roll angle of the gap between 0° and 45° and the pitch angle between 0° and 30°. We do this for two reasons. First, when the gap is heavily pitched, the quadrotor needs more space to reach the initial conditions of the traverse from hover. This renders the gap barely or not visible at the start of the approach, increasing the uncertainty in the pose estimation. Second, extreme configurations, such as roll angles of the gap up to 90°, require high angular velocities in order to let the quadrotor align its orientation with that of the gap. This makes gap detection difficult, if not impossible, due to motion blur. Also, our current experimental setup does not allow us to apply the torques necessary to reach high angular velocities because of the inertia of the platform and motor saturations.

E.5.4 Dealing with Missing Gap Detections

The algorithm proposed in Sec. E.3.1 fuses the poses from gap detection with IMU readings to provide the full state estimate during the approach maneuver. In case of motion blur, due to high angular velocities, or when the vehicle is too close to the gap, the gap detection algorithm does not return any pose estimate. However, these situations do not represent an issue during short periods of time (a few tenths of a second). In these cases, the state estimate from the sensor fusion module is still available and reliable through the IMU.

E.6 Conclusion

We developed a system that lets a quadrotor vehicle safely pass through a narrow inclined gap using only onboard sensing and computing. Full state estimation is provided by fusing gap detections from a forward-facing onboard camera and an IMU.

To tackle the problems arising from the varying uncertainty from the vision-based state estimation, we coupled perception and control by computing trajectories that facilitate state estimation by always keeping the gap in the image of the onboard camera.

We successfully evaluated and demonstrated the approach in many real-world experiments. To the best of our knowledge, this is the first work that addresses and achieves autonomous, aggressive flight through narrow gaps using only onboard sensing and computing, and without requiring prior knowledge of the pose of the gap. We believe that this is a major step forward autonomous quadrotor flight in complex environments with onboard sensing and computing.

F Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag

M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 620–626. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2776353

Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories

Matthias Faessler, Antonio Franchi, and Davide Scaramuzza

Abstract — In this paper, we prove that the dynamical model of a quadrotor subject to linear rotor drag effects is differentially flat in its position and heading. We use this property to compute feed-forward control terms directly from a reference trajectory to be tracked. The obtained feed-forward terms are then used in a cascaded, nonlinear feedback control law that enables accurate agile flight with quadrotors. Compared to state-of-the-art control methods, which treat the rotor drag as an unknown disturbance, our method reduces the trajectory tracking error significantly. Finally, we present a method based on a gradient-free optimization to identify the rotor drag coefficients, which are required to compute the feed-forward control terms. The new theoretical results are thoroughly validated trough extensive comparative experiments.

Supplementary Material

Video of the experiments: https://youtu.be/VIQILwcM5PA

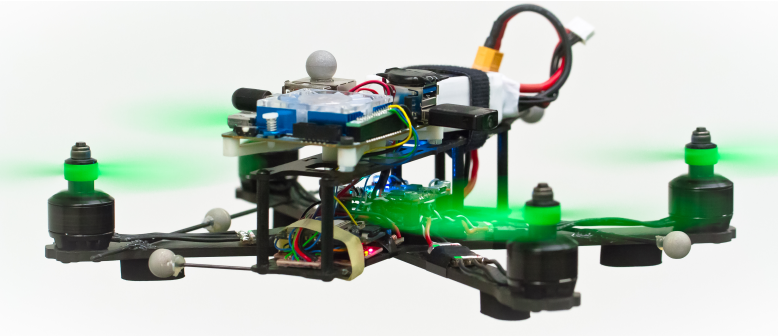


Figure F.1: First-person-view racing inspired quadrotor platform used for the presented experiments.

F.1 Introduction

F.1.1 Motivation

For several years, quadrotors have proven to be suitable aerial platforms for performing agile flight maneuvers . Nevertheless, quadrotors are typically controlled by neglecting aerodynamic effects, such as rotor drag, that only become important for non-hover conditions. These aerodynamic effects are treated as unknown disturbances, which works well when controlling the quadrotor close to hover conditions but reduces its trajectory tracking accuracy progressively with increasing speed. For fast obstacle avoidance it is important to perform accurate agile trajectory tracking. To achieve this, we require a method for accurate tracking of trajectories that are unknown prior to flying.

The main aerodynamic effect causing trajectory tracking errors during high-speed flight is rotor drag, which is a linear effect in the quadrotor's velocity [24]. In this work, we aim at developing a control method that improves the trajectory tracking performance of quadrotors by considering the rotor drag effect. To achieve this, we first prove that the dynamical model of a quadrotor subject to linear rotor drag effects is differentially flat with flat outputs chosen to be its position and heading. We then use this property to compute feed-forward control terms directly from the reference trajectory to be tracked. The obtained feed-forward terms are then used in a cascaded, nonlinear feedback control law that enables accurate agile flight with quadrotors on a priori unknown trajectories. Finally, we present a method based on a gradient-free optimization to identify the rotor drag coefficients which are required to compute the feed-forward control terms. We validate our theoretical results through experiments with a quadrotor shown in Fig. F.1.

F.1.2 Related Work

In [106], it was shown that the common model of a quadrotor *without* considering rotor drag effects is differentially flat when choosing its position and heading as flat outputs. Furthermore, this work presented a control algorithm that computes the desired collective thrust and torque inputs from the measured position, velocity, orientation, and body-rates errors. With this method, agile maneuvers with speeds of several meters per second were achieved. In [42], the differential flatness property of a hexarotor that takes the desired collective thrust and its desired orientation as inputs was exploited to compute feed-forward terms used in an LQR feedback controller. The desired orientation was then controlled by a separate low-level control loop, which also enables the execution of flight maneuvers with speeds of several meters per second. We extend these works by showing that the dynamics of a quadrotor are differentially flat even when they are subject to linear rotor drag effects. Similarly to [42], we make use of this property to compute feed-forward terms that are then applied by a position controller.

Rotor drag effects influencing a quadrotor's dynamics were investigated in [22] and [102] where also a control law was presented, which considers these dynamics. Rotor drag effects originate from blade flapping and induced drag of the rotors, which are, thanks to their equivalent mathematical expression, typically combined as linear effects in a lumped parameter dynamical model [100]. These rotor drag effects were then incorporated in dynamical models of multi rotors to improve state estimation in [90] and [25]. In this work, we make use of the fact that the main aerodynamic effects are of similar nature and can therefore be described together by lumped parameters in a dynamical model.

In [10], the authors achieve accurate thrust control by electronic speed controllers through a model of the aerodynamic power generated by a fixed-pitch rotor under wind disturbances, which reduces the trajectory tracking error of a quadrotor. Rotor drag was also considered in control methods for multi-rotor vehicles in [76] and [127], where the control problem was simplified by decomposing the rotor drag force into a component that is independent of the vehicle's orientation and one along the thrust direction, which leads to an explicit expression for the desired thrust direction. In [161], a refined thrust model and a control scheme that considers rotor drag in the computation of the thrust command and the desired orientation are presented. Additionally to the thrust command and desired orientation, the control scheme in [8] also computes the desired body rates and angular accelerations by considering rotor drag but requires estimates of the quadrotor's acceleration and jerk, which are usually not available. In contrast, we compute the exact reference thrust, orientation, body rates, and angular accelerations considering rotor drag only from a reference trajectory, which we then use as feed-forward terms in the controller.

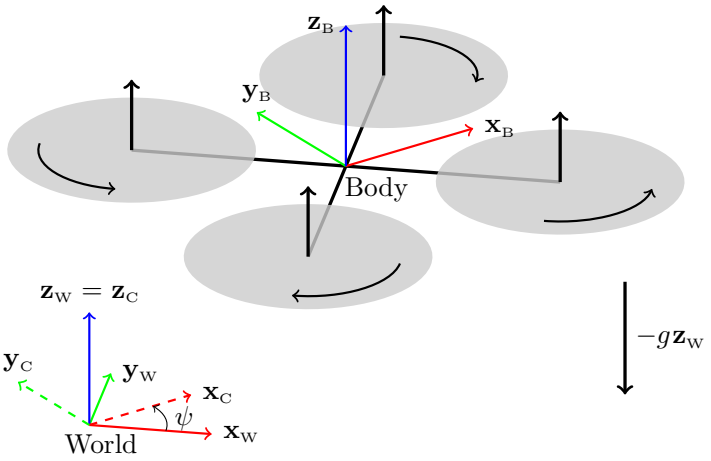


Figure F.2: Schematics of the considered quadrotor model with the used coordinate systems.

F.2 Nomenclature

In this work, we make use of a world frame W with orthonormal basis $\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$ represented in world coordinates and a body frame B with orthonormal basis $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$ also represented in world coordinates. The body frame is fixed to the quadrotor with an origin coinciding with its center of mass as depicted in Fig. F.2. The quadrotor is subject to a gravitational acceleration g in the $-\mathbf{z}_W$ direction. We denote the position of the quadrotor's center of mass as $\mathbf{p}$, and its derivatives, velocity, acceleration, jerk, and snap as $\mathbf{v}$, $\mathbf{a}$, $\mathbf{j}$, and $\mathbf{s}$, respectively. We represent the quadrotor's orientation as a rotation matrix $\mathbf{R} = \begin{bmatrix}\mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B\end{bmatrix}$ and its body rates (i.e., the angular velocity) as $\boldsymbol{\omega}$ represented in body coordinates. To denote a unit vector along the z-coordinate axis we write $\mathbf{e}_z$. Finally, we denote quantities that can be computed from a reference trajectory as *reference values* and quantities that are computed by an outer loop feedback control law and passed to an inner loop controller as *desired values*.

F.3 Model

We consider the dynamical model of a quadrotor with rotor drag developed in [76] with no wind, stiff propellers, and no dependence of the rotor drag on the thrust. According to this model, the dynamics of the position $\mathbf{p}$, velocity $\mathbf{v}$, orientation $\mathbf{R}$, and body rates $\boldsymbol{\omega}$ can be written as

$$\dot{\mathbf{p}} = \mathbf{v} \tag{F.1}$$

$$\dot{\mathbf{v}} = -g\mathbf{z}_W + c\mathbf{z}_B - \mathbf{R}\mathbf{D}\mathbf{R}^\top\mathbf{v} \tag{F.2}$$

$$\dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}} \tag{F.3}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} - \boldsymbol{\tau}_g - \mathbf{A}\mathbf{R}^\top\mathbf{v} - \mathbf{B}\boldsymbol{\omega}\right) \tag{F.4}$$

where c is the mass-normalized collective thrust, $\mathbf{D} = \text{diag}\,(d_x, d_y, d_z)$ is a constant diagonal matrix formed by the mass-normalized rotor-drag coefficients, $\hat{\boldsymbol{\omega}}$ is a skew-symmetric matrix formed from $\boldsymbol{\omega}$, $\mathbf{J}$ is the quadrotor's inertia matrix, $\boldsymbol{\tau}$ is the three dimensional torque input, $\boldsymbol{\tau}_g$ are gyroscopic torques from the propellers, and $\mathbf{A}$ and $\mathbf{B}$ are constant matrices. For the derivations and more details about these terms, please refer to [76]. In this work, we adopt the thrust model presented in [161]

$$c = c_{\text{cmd}} + k_h v_h^2 \tag{F.5}$$

where c_{cmd} is the commanded collective thrust input, k_h is a constant, and $v_h = \mathbf{v}^\top(\mathbf{x}_B + \mathbf{y}_B)$. The term $k_h v_h^2$ acts as a quadratic velocity-dependent input disturbance which adds up to the input c_{cmd}. The additional linear velocity-dependent disturbance in the $\mathbf{z}_B$ direction of the thrust model in [161] is lumped by d_z directly in (F.2) by neglecting its dependency on the rotor speeds. Note that this dynamical model of a quadrotor is a generalization of the common model found, e.g., in [106], in which the linear rotor drag components are typically neglected, i.e., $\mathbf{D}$, $\mathbf{A}$ and $\mathbf{B}$ are considered null matrices.

F.4 Differential Flatness

In this section, we show that the extended dynamical model of a quadrotor subject to rotor drag (F.1)-(F.4) with four inputs is differentially flat, like the model with neglected drag [106]. In fact, we shall show that the states $[\mathbf{p}, \mathbf{v}, \mathbf{R}, \boldsymbol{\omega}]$ and the inputs $[c_{\text{cmd}}, \boldsymbol{\tau}]$ can be written as algebraic functions of four selected flat outputs and a finite number of their derivatives. Equally to [106], we choose the flat outputs to be the quadrotor's position $\mathbf{p}$ and its heading ψ.

To show that the orientation $\mathbf{R}$ and the collective thrust c are functions of the flat outputs, we reformulate (F.2) as

$$\begin{aligned} &c\mathbf{z}_B - (d_x \mathbf{x}_B^\top \mathbf{v})\,\mathbf{x}_B - (d_y \mathbf{y}_B^\top \mathbf{v})\,\mathbf{y}_B - (d_z \mathbf{z}_B^\top \mathbf{v})\,\mathbf{z}_B \\ &\qquad - \mathbf{a} - g\mathbf{z}_W = 0. \end{aligned} \tag{F.6}$$

From left-multiplying (F.6) by $\mathbf{x}_B^\top$ we get

$$\mathbf{x}_B^\top \boldsymbol{\alpha} = 0, \text{ with } \quad \boldsymbol{\alpha} = \mathbf{a} + g\mathbf{z}_W + d_x \mathbf{v}. \tag{F.7}$$

From left-multiplying (F.6) by $\mathbf{y}_B^\top$ we get

$$\mathbf{y}_B^\top \boldsymbol{\beta} = 0, \text{ with } \quad \boldsymbol{\beta} = \mathbf{a} + g\mathbf{z}_W + d_y \mathbf{v}. \tag{F.8}$$

To enforce a reference heading ψ, we constrain the projection of the $\mathbf{x}_B$ axis into the

$\mathbf{x}_W - \mathbf{y}_W$ plane to be collinear with $\mathbf{x}_C$ (cf. Fig. F.2), where

$$\mathbf{x}_C = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \end{bmatrix}^\top \tag{F.9}$$

$$\mathbf{y}_C = \begin{bmatrix} -\sin(\psi) & \cos(\psi) & 0 \end{bmatrix}^\top. \tag{F.10}$$

From this, (F.7) and (F.8), and the constraints that $\mathbf{x}_B$, $\mathbf{y}_B$, and $\mathbf{z}_B$ must be orthogonal to each other and of unit length, we can construct $\mathbf{R}$ with

$$\mathbf{x}_B = \frac{\mathbf{y}_C \times \boldsymbol{\alpha}}{\|\mathbf{y}_C \times \boldsymbol{\alpha}\|} \tag{F.11}$$

$$\mathbf{y}_B = \frac{\boldsymbol{\beta} \times \mathbf{x}_B}{\|\boldsymbol{\beta} \times \mathbf{x}_B\|} \tag{F.12}$$

$$\mathbf{z}_B = \mathbf{x}_B \times \mathbf{y}_B. \tag{F.13}$$

One can verify that these vectors are of unit length, perpendicular to each other, and satisfy the constraints (F.7) - (F.10). To get the collective thrust, we left-multiply (F.6) by $\mathbf{z}_B^\top$

$$c = \mathbf{z}_B^\top \left(\mathbf{a} + g\mathbf{z}_W + d_z \mathbf{v} \right). \tag{F.14}$$

Then the collective thrust input can be computed as a function of c, $\mathbf{R}$, and the flat outputs, as

$$c_{\text{cmd}} = c - k_h (\mathbf{v}^\top (\mathbf{x}_B + \mathbf{y}_B))^2. \tag{F.15}$$

To show that the body rates $\boldsymbol{\omega}$ are functions of the flat outputs and their derivatives, we take the derivative of (F.2)

$$\mathbf{j} = \dot{c}\mathbf{z}_B + c\mathbf{R}\hat{\boldsymbol{\omega}}\mathbf{e}_z - \mathbf{R}\left((\hat{\boldsymbol{\omega}}\mathbf{D} + \mathbf{D}\hat{\boldsymbol{\omega}}^\top)\, \mathbf{R}^\top \mathbf{v} + \mathbf{D}\mathbf{R}^\top \mathbf{a} \right). \tag{F.16}$$

Left-multiplying (F.16) by $\mathbf{x}_B^\top$ and rearranging terms, we get

$$\begin{aligned} &\omega_y \left(c - (d_z - d_x) \left(\mathbf{z}_B^\top \mathbf{v} \right) \right) - \omega_z \left(d_x - d_y \right) \left(\mathbf{y}_B^\top \mathbf{v} \right) \\ &\quad = \mathbf{x}_B^\top \mathbf{j} + d_x \mathbf{x}_B^\top \mathbf{a}. \end{aligned} \tag{F.17}$$

Left-multiplying (F.16) by $\mathbf{y}_B^\top$ and rearranging terms, we get

$$\begin{aligned} &\omega_x \left(c + (d_y - d_z) \left(\mathbf{z}_B^\top \mathbf{v} \right) \right) + \omega_z \left(d_x - d_y \right) \left(\mathbf{x}_B^\top \mathbf{v} \right) \\ &\quad = -\mathbf{y}_B^\top \mathbf{j} - d_y \mathbf{y}_B^\top \mathbf{a}. \end{aligned} \tag{F.18}$$

To get a third constraint for the body rates, we project (F.3) along $\mathbf{y}_B$

$$\omega_z = \mathbf{y}_B^\top \dot{\mathbf{x}}_B. \tag{F.19}$$

Since $\mathbf{x}_B$ is perpendicular to $\mathbf{y}_C$ and $\mathbf{z}_B$, we can write

$$\mathbf{x}_B = \frac{\tilde{\mathbf{x}}_B}{\|\tilde{\mathbf{x}}_B\|}, \text{ with } \quad \tilde{\mathbf{x}}_B = \mathbf{y}_C \times \mathbf{z}_B. \tag{F.20}$$

Taking its derivative as the general derivative of a normalized vector, we get

$$\dot{\mathbf{x}}_B = \frac{\dot{\tilde{\mathbf{x}}}_B}{\|\tilde{\mathbf{x}}_B\|} - \tilde{\mathbf{x}}_B \frac{\tilde{\mathbf{x}}_B^\top \dot{\tilde{\mathbf{x}}}_B}{\|\tilde{\mathbf{x}}_B\|^3} \tag{F.21}$$

and, since $\tilde{\mathbf{x}}_B$ is collinear to $\mathbf{x}_B$ and therefore perpendicular to $\mathbf{y}_B$, we can write (F.19) as

$$\omega_z = \mathbf{y}_B^\top \frac{\dot{\tilde{\mathbf{x}}}_B}{\|\tilde{\mathbf{x}}_B\|}. \tag{F.22}$$

The derivative of $\tilde{\mathbf{x}}_B$ can be computed as

$$\dot{\tilde{\mathbf{x}}}_B = \dot{\mathbf{y}}_C \times \mathbf{z}_B + \mathbf{y}_C \times \dot{\mathbf{z}}_B, \tag{F.23}$$

$$= (-\dot{\psi}\mathbf{x}_C) \times \mathbf{z}_B + \mathbf{y}_C \times \left(\omega_y \mathbf{x}_B - \omega_x \mathbf{y}_B\right). \tag{F.24}$$

From this, (F.20), (F.22), and the vector triple product $\mathbf{a}^\top(\mathbf{b} \times \mathbf{c}) = -\mathbf{b}^\top(\mathbf{a} \times \mathbf{c})$ we then get

$$\omega_z = \frac{1}{\|\mathbf{y}_C \times \mathbf{z}_B\|} \left(\dot{\psi}\mathbf{x}_C^\top \mathbf{x}_B + \omega_y \mathbf{y}_C^\top \mathbf{z}_B\right). \tag{F.25}$$

The body rates can now be obtained by solving the linear system of equations composed of (F.17), (F.18), and (F.25).

To compute the angular accelerations $\dot{\boldsymbol{\omega}}$ as functions of the flat outputs and their derivatives, we take the derivative of (F.17), (F.18), and (F.25) to get a similar linear system of equations as

$$\begin{aligned}
&\dot{\omega}_y \left(c - (d_z - d_x)\left(\mathbf{z}_B^\top \mathbf{v}\right)\right) - \dot{\omega}_z \left(d_x - d_y\right)\left(\mathbf{y}_B^\top \mathbf{v}\right) \\
&\quad = \mathbf{x}_B^\top \mathbf{s} - 2\dot{c}\omega_y - c\omega_x\omega_z + \mathbf{x}_B^\top \boldsymbol{\xi} \qquad \text{(F.26)} \\
&\dot{\omega}_x \left(c + (d_y - d_z)\left(\mathbf{z}_B^\top \mathbf{v}\right)\right) + \dot{\omega}_z \left(d_x - d_y\right)\left(\mathbf{x}_B^\top \mathbf{v}\right) \\
&\quad = -\mathbf{y}_B^\top \mathbf{s} - 2\dot{c}\omega_x + c\omega_y\omega_z - \mathbf{y}_B^\top \boldsymbol{\xi} \qquad \text{(F.27)} \\
&-\dot{\omega}_y \mathbf{y}_C^\top \mathbf{z}_B + \dot{\omega}_z \|\mathbf{y}_C \times \mathbf{z}_B\| \\
&\quad = \ddot{\psi}\mathbf{x}_C^\top \mathbf{x}_B + 2\dot{\psi}\omega_z \mathbf{x}_C^\top \mathbf{y}_B - 2\dot{\psi}\omega_y \mathbf{x}_C^\top \mathbf{z}_B \\
&\qquad - \omega_x\omega_y \mathbf{y}_C^\top \mathbf{y}_B - \omega_x\omega_z \mathbf{y}_C^\top \mathbf{z}_B \qquad \text{(F.28)}
\end{aligned}$$

which we can solve for $\dot{\omega}$ with

$$\begin{aligned}\dot{c} &= \mathbf{z}_B^\top \mathbf{j} + \omega_x \left(d_y - d_z\right) \left(\mathbf{y}_B^\top \mathbf{v}\right) \\ &\quad + \omega_y \left(d_z - d_x\right) \left(\mathbf{x}_B^\top \mathbf{v}\right) + d_z \mathbf{z}_B^\top \mathbf{a} \qquad \text{(F.29)} \\ \xi &= \mathbf{R} \left(\hat{\omega}^2 \mathbf{D} + \mathbf{D}\hat{\omega}^2 + 2\hat{\omega}\mathbf{D}\hat{\omega}^\top\right) \mathbf{R}^\top \mathbf{v} \\ &\quad + 2\mathbf{R} \left(\hat{\omega}\mathbf{D} + \mathbf{D}\hat{\omega}^\top\right) \mathbf{R}^\top \mathbf{a} + \mathbf{R}\mathbf{D}\mathbf{R}^\top \mathbf{j}. \qquad \text{(F.30)}\end{aligned}$$

Once we know the angular accelerations, we can solve (F.4) for the torque inputs $\boldsymbol{\tau}$.

Note that, besides quadrotors, this proof also applies to multi-rotor vehicles with parallel rotor axes in general. More details of this proof can be found in our technical report [38].

F.5 Control Law

To track a reference trajectory, we use a controller consisting of feedback terms computed from tracking errors as well as feed-forward terms computed from the reference trajectory using the quadrotor's differential flatness property. Apart from special cases, the control architectures of typical quadrotors do not allow to apply the torque inputs directly. They instead provide a low-level body-rate controller, which accepts desired body rates. In order to account for this possibility, we designed our control algorithm with a classical cascaded structure, i.e., consisting of a high-level position controller and the low-level body-rate controller. The high-level position controller computes the desired orientation $\mathbf{R}_{\text{des}}$, the collective thrust input c_{cmd}, the desired body rates $\boldsymbol{\omega}_{\text{des}}$, and the desired angular accelerations $\dot{\boldsymbol{\omega}}_{\text{des}}$, which are then applied in a low-level controller (e.g. as presented in [35]). As a first step in the position controller, we compute the desired acceleration of the quadrotor's body as

$$\mathbf{a}_{\text{des}} = \mathbf{a}_{\text{fb}} + \mathbf{a}_{\text{ref}} - \mathbf{a}_{\text{rd}} + g\mathbf{z}_W \qquad \text{(F.31)}$$

where $\mathbf{a}_{\text{fb}}$ are the PD feedback-control terms computed from the position and velocity control errors as

$$\mathbf{a}_{\text{fb}} = -\mathbf{K}_{\text{pos}} \left(\mathbf{p} - \mathbf{p}_{\text{ref}}\right) - \mathbf{K}_{\text{vel}} \left(\mathbf{v} - \mathbf{v}_{\text{ref}}\right) \qquad \text{(F.32)}$$

where $\mathbf{K}_{\text{pos}}$ and $\mathbf{K}_{\text{vel}}$ are constant diagonal matrices and $\mathbf{a}_{\text{rd}} = -\mathbf{R}_{\text{ref}}\mathbf{D}\mathbf{R}_{\text{ref}}^\top \mathbf{v}_{\text{ref}}$ are the accelerations due to rotor drag. We compute the desired orientation $\mathbf{R}_{\text{des}}$ such that the

desired acceleration $\mathbf{a}_{\text{des}}$ and the reference heading ψ_{ref} is respected as

$$\mathbf{z}_{\text{B,des}} = \frac{\mathbf{a}_{\text{des}}}{\|\mathbf{a}_{\text{des}}\|} \tag{F.33}$$

$$\mathbf{x}_{\text{B,des}} = \frac{\mathbf{y}_{\text{C}} \times \mathbf{z}_{\text{B,des}}}{\|\mathbf{y}_{\text{C}} \times \mathbf{z}_{\text{B,des}}\|} \tag{F.34}$$

$$\mathbf{y}_{\text{B,des}} = \mathbf{z}_{\text{B,des}} \times \mathbf{x}_{\text{B,des}}. \tag{F.35}$$

By projecting the desired accelerations onto the actual body z-axis and considering the thrust model (F.5), we can then compute the collective thrust input as

$$c_{\text{cmd}} = \mathbf{a}_{\text{des}}^{\top} \mathbf{z}_{\text{B}} - k_h (\mathbf{v}^{\top} (\mathbf{x}_{\text{B}} + \mathbf{y}_{\text{B}}))^2. \tag{F.36}$$

Similarly, we can compute the desired body rates as

$$\boldsymbol{\omega}_{\text{des}} = \boldsymbol{\omega}_{\text{fb}} + \boldsymbol{\omega}_{\text{ref}} \tag{F.37}$$

where $\boldsymbol{\omega}_{\text{fb}}$ are the feedback terms computed from an attitude controller (e.g. as presented in [37]) and $\boldsymbol{\omega}_{\text{ref}}$ are feed-forward terms from the reference trajectory, which are computed as described in Section F.4. Finally, the desired angular accelerations are the reference angular accelerations

$$\dot{\boldsymbol{\omega}}_{\text{des}} = \dot{\boldsymbol{\omega}}_{\text{ref}} \tag{F.38}$$

which are computed from the reference trajectory as described in Section F.4.

F.6 Drag Coefficients Estimation

To apply the presented control law with inputs $[c_{\text{cmd}}, \boldsymbol{\omega}]$, we need to identify $\mathbf{D}$, and k_h, which are used to compute the reference inputs and the thrust command. If instead one is using a platform that is controlled by the inputs $[c_{\text{cmd}}, \boldsymbol{\tau}]$, $\mathbf{A}$ and $\mathbf{B}$ also need to be identified for computing the torque input. While $\mathbf{D}$ and k_h can be accurately estimated from measured accelerations and velocities through (F.2) and (F.5), the effects of $\mathbf{A}$ and $\mathbf{B}$ on the body-rate dynamics (F.4) are weaker and require to differentiate the gyro measurements as well as knowing the rotor speeds and rotor inertia to compute $\boldsymbol{\tau}$ and $\boldsymbol{\tau}_{\text{g}}$. Therefore, we propose to identify $\mathbf{D}$, $\mathbf{A}$, $\mathbf{B}$, and k_h by running a Nelder-Mead gradient free optimization [123] for which the quadrotor repeats a predefined trajectory in each iteration of the optimization. During this procedure, we control the quadrotor by the proposed control scheme with different drag coefficients in each iteration during which we record the absolute trajectory tracking error (F.39) and use it as cost for the optimization. Once the optimization has converged, we know the coefficients that reduce the trajectory tracking error the most. We found that the obtained values for $\mathbf{D}$ agree with an estimation through (F.2) when recording IMU measurements and

ground truth velocity. This procedure has the advantage that no IMU and rotor speed measurements are required, which are both unavailable on our quadrotor platform used for the presented experiments, and the gyro measurements do not need to be differentiated. This is not the case when performing the identification through (F.2), (F.4), and (F.5). Also, since our method does not rely explicitly on (F.2), it can also capture first order approximations of non modeled effects lumped into the identified coefficients. Furthermore, our implementation allows stopping and restarting the optimization at any time, which allows changing the battery.

F.7 Experiments

F.7.1 Experimental Setup

Our quadrotor platform is built from off-the-shelf components used for first-person-view racing (see Fig. F.1). It features a carbon frame with stiff six inch propellers, a Raceflight Revolt flight controller, an Odroid XU4 single board computer, and a Laird RM024 radio module for receiving control commands. The platform weights 610 g and has a thrust-to-weight ratio of 4. To improve its trajectory tracking accuracy we compensate the thrust commands for the varying battery voltage. All the presented flight experiments were conducted in an OptiTrack motion capture system to acquire the ground truth state of the quadrotor which is obtained at 200 Hz and is used for control and evaluation of the trajectory tracking performance. Note that our control method and the rotor-drag coefficient identification also work with state estimates that are obtained differently than with a motion capture system. We compensate for an average latency of the perception and control pipeline of 32 ms. The high-level control runs on a laptop computer at 55 Hz, sending collective thrust and body rate commands to the on-board flight controller where they are tracked by a PD controller running at 4 kHz.

To evaluate the trajectory tracking performance and as cost for estimating the drag coefficients, we use the absolute trajectory tracking error defined as the root mean square position error

$$E_a = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \|E_p^k\|^2}, \text{ where } \quad E_p^k = \mathbf{p}^{\mathrm{k}} - \mathbf{p}_{\mathrm{ref}}^{\mathrm{k}} \tag{F.39}$$

over N control cycles of the high-level controller required to execute a given trajectory.

F.7.2 Trajectories

For identifying the rotor-drag coefficients and for demonstrating the trajectory tracking performance of the proposed controller, we let the quadrotor execute a horizontal circle trajectory and a horizontal Gerono lemniscate trajectory. The circle trajectory has a radius of 1.8 m with a velocity of $4\,\mathrm{m\,s^{-1}}$ resulting, for the case of not considering rotor drag, in a required collective mass-normalized thrust of $c = 13.24\,\mathrm{m\,s^{-2}}$ and a maximum body rates norm of $\|\boldsymbol{\omega}\| = 85\,^\circ\,\mathrm{s^{-1}}$. Its maximum nominal velocity in the $\mathbf{x}_\mathrm{B}$ and $\mathbf{y}_\mathrm{B}$ is $4.0\,\mathrm{m\,s^{-1}}$ and $0.0\,\mathrm{m\,s^{-1}}$ in the $\mathbf{z}_\mathrm{B}$ direction. The Gerono lemniscate trajectory is defined by $\left[x(t) = 2\cos\left(\sqrt{2}t\right); y(t) = 2\sin\left(\sqrt{2}t\right)\cos\left(\sqrt{2}t\right)\right]$ with a maximum velocity of $4\,\mathrm{m\,s^{-1}}$, a maximum collective mass-normalized thrust of $c = 12.98\,\mathrm{m\,s^{-2}}$, and a maximum body rates norm of $\|\boldsymbol{\omega}\| = 136\,^\circ\,\mathrm{s^{-1}}$. Its maximum nominal velocity in the $\mathbf{x}_\mathrm{B}$ and $\mathbf{y}_\mathrm{B}$ is $2.8\,\mathrm{m\,s^{-1}}$ and $1.3\,\mathrm{m\,s^{-1}}$ in the $\mathbf{z}_\mathrm{B}$ direction for the case of not considering rotor drag.

F.7.3 Drag Coefficients Identification

To identify the drag coefficients in $\mathbf{D}$, we ran the optimization presented in Section F.6 multiple times on both the circle and lemniscate trajectories until it converged, i.e., until the changes of each coefficient in one iteration is below a specified threshold. We do not identify $\mathbf{A}$ and $\mathbf{B}$ since the quadrotor platform used for the presented experiments takes $[c_\mathrm{cmd}, \boldsymbol{\omega}]$ as inputs and we therefore do not need to compute the torque inputs involving $\mathbf{A}$ and $\mathbf{B}$ according to (F.4). Since we found that d_z and k_h only have minor effects on the trajectory tracking performance, we isolate the effects of d_x and d_y by setting $d_z = 0$ and $k_h = 0$ for the presented experiments. For each iteration of the optimization, the quadrotor flies two loops of either trajectory. The optimization typically converges after about 70 iterations, which take around 30 min including multiple battery swaps.

The evolution of the best performing drag coefficients is shown in Fig. F.3 for every iteration of the proposed optimization. On the circle trajectory, we obtained $d_\mathrm{x} = 0.544\,\mathrm{s^{-1}}$ and $d_\mathrm{y} = 0.386\,\mathrm{s^{-1}}$, whereas on the lemniscate trajectory we obtained $d_\mathrm{x} = 0.491\,\mathrm{s^{-1}}$ and $d_\mathrm{y} = 0.236\,\mathrm{s^{-1}}$. For both trajectories, d_x is larger than d_y, which is expected because we use a quadrotor that is wider than long. The obtained drag coefficients identified on the circle are different than the ones identified on the lemniscate, which is due to the fact that the circle trajectory excites velocities in the $\mathbf{x}_\mathrm{B}$ and $\mathbf{y}_\mathrm{B}$ more than the lemniscate trajectory. We could verify this claim by running the identification on the circle trajectory with a speed of $2.8\,\mathrm{m\,s^{-1}}$, which corresponds to the maximum speeds reached in $\mathbf{x}_\mathrm{B}$ and $\mathbf{y}_\mathrm{B}$ on the lemniscate trajectory. For this speed, we obtained $d_\mathrm{x} = 0.425\,\mathrm{s^{-1}}$, and $d_\mathrm{y} = 0.256\,\mathrm{s^{-1}}$ on the circle trajectory, which are close to the coefficients identified on the lemniscate trajectory. Additionally, in our dynamical model, we assume the rotor drag to be independent of the thrust. This is not true in reality and therefore leads to different results of the drag coefficient estimation on different trajectories where

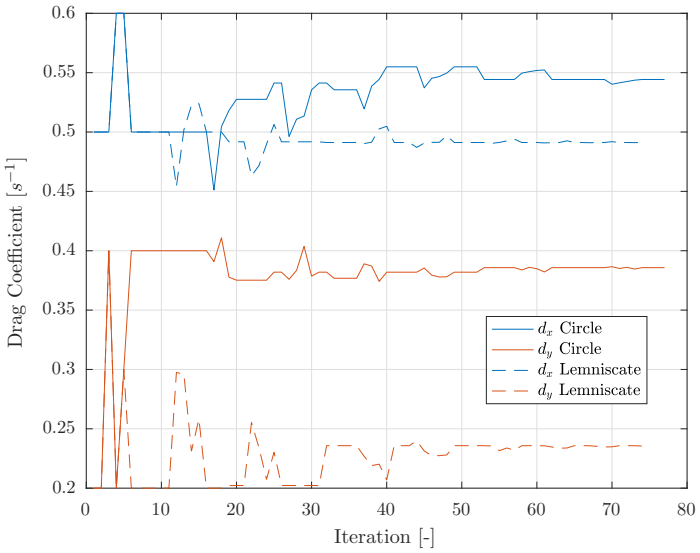


Figure F.3: The best performing drag coefficients d_x and d_y for every iteration of the identification on both the circle and the lemniscate trajectory.

different thrusts are applied. These reasons suggest to carefully select a trajectory for the identification, which goes towards the problem of finding the optimal trajectory for parameter estimation, which is outside the scope of this paper. In all the conducted experiments, we found that a non zero drag coefficient in the z-direction d_z does not improve the trajectory tracking performance. We found this to be true even for purely vertical trajectories with velocities of up to $2.5\,\mathrm{m\,s^{-1}}$. Furthermore, we found that an estimated $k_h = 0.009\,\mathrm{m}^{-1}$ improves the trajectory tracking performance further but by about one order of magnitude less than d_x and d_y on the considered trajectories.

F.7.4 Trajectory Tracking Performance

To demonstrate the trajectory tracking performance of the proposed control scheme, we compare the position error of our quadrotor flying the circle and the lemniscate trajectory described above for three conditions: (i) without considering rotor drag, (ii) with the drag coefficients estimated on the circle trajectory, and (iii) with the drag coefficients estimated on the lemniscate trajectory.[1] Fig. F.4 shows the ground truth and reference position when flying the circle trajectory under these three conditions. Equally, Fig. F.5 shows the ground truth and reference position when flying the lemniscate trajectory under the same three conditions. The tracking performance statistics for both trajectories are summarized in Table F.1. From these statistics, we see that the trajectory tracking performance has improved significantly when considering rotor drag on both

[1] Video of the experiments: https://youtu.be/VIQILwcM5PA

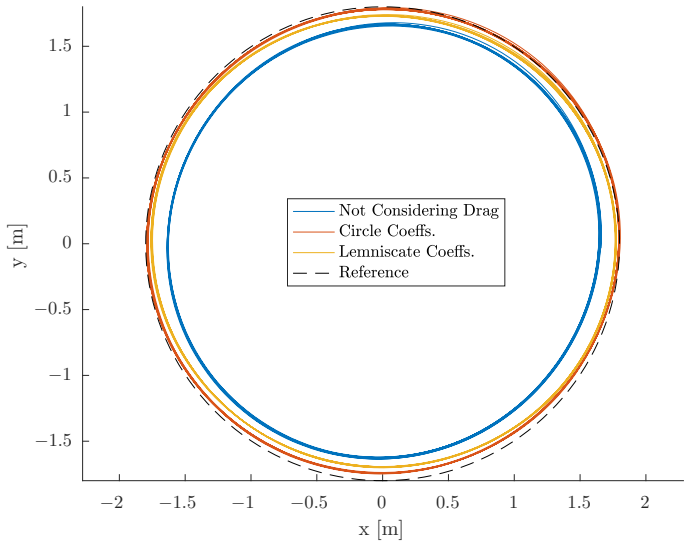


Figure F.4: Ground truth position for ten loops on the circle trajectory without considering rotor drag (solid blue), with drag coefficients estimated on the circle trajectory (solid red), and with drag coefficients estimated on the lemniscate trajectory (solid yellow) compared to the reference position (dashed black).

trajectories independently of which trajectory the rotor-drag coefficients were estimated on. This confirms that our approach is applicable to any feasible trajectory once the drag coefficients are identified, which is an advantage over methods that improve tracking performance for a specific trajectory only (e.g. [64]). With the rotor-drag coefficients estimated on the circle trajectory, we achieve almost the same performance in terms of absolute trajectory tracking error on the lemniscate trajectory as with the coefficients identified on the lemniscate trajectory but not vice versa. As discussed above, this is due to a higher excitation in body velocities on the circle trajectory which results in a better identification of the rotor-drag coefficients. This suggests to perform the rotor-drag coefficients identification on a trajectory that maximally excites the body velocities.

Since the rotor drag is a function of the velocity of the quadrotor, we show the benefits of our control approach by linearly ramping up the maximum speed on both trajectories from $0\,\mathrm{m\,s^{-1}}$ to $5\,\mathrm{m\,s^{-1}}$ in 30 s. Fig. F.6, and Fig. F.7 show the position error norm and the reference speed over time until the desired maximum speed of $5\,\mathrm{m\,s^{-1}}$ is reached for the circle and the lemniscate trajectory, respectively. Both figures show that considering rotor drag does not improve trajectory tracking for small speeds below $0.5\,\mathrm{m\,s^{-1}}$ but noticeably does so for higher speeds.

An analysis of the remaining position error for the case where rotor drag is considered reveals that it strongly correlates to the applied collective thrust. In the used dynamical

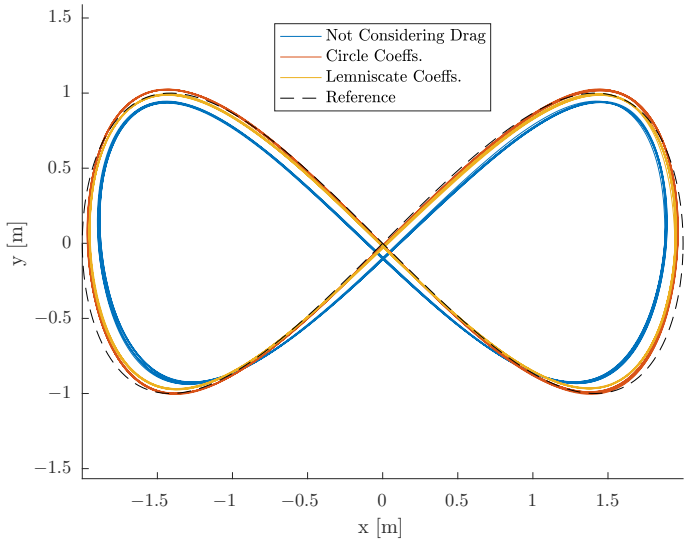


Figure F.5: Ground truth position for ten loops on the lemniscate trajectory without considering rotor drag (solid blue), with drag coefficients estimated on the circle trajectory (solid red), and with drag coefficients estimated on the lemniscate trajectory (solid yellow) compared to the reference position (dashed black).

model of a quadrotor, we assume the rotor drag to be independent of the thrust [c.f. (F.2)], which is not true in reality. For the experiment in Fig. F.6, the commanded mass-normalized collective thrust varies between $10\,\mathrm{m\,s^{-2}}$ and $18\,\mathrm{m\,s^{-2}}$ which clearly violates the constant thrust assumption. By considering a dependency of the rotor drag on the thrust might improve trajectory tracking even further and is subject of future work.

F.8 Comparison to Other Control Methods

In this section, we present a qualitative comparison to other quadrotor controllers that consider rotor drag effects as presented in [76], [127], [161], and [8].

None of these works show or exploit the differential flatness property of quadrotor dynamics subject to rotor drag effects. They also do not consider asymmetric vehicles where $d_x \neq d_y$ and they omit the computation of ω_z and $\dot{\omega}_z$.

In [76] and [127], the presented position controller decomposes the rotor drag force into a component that is independent of the vehicle's orientation and one along the thrust direction, which leads to an explicit expression for the desired thrust direction. They both neglect feed-forward on angular accelerations, which does not allow perfect trajectory tracking. As in our work, [76] models the rotor drag to be proportional to the

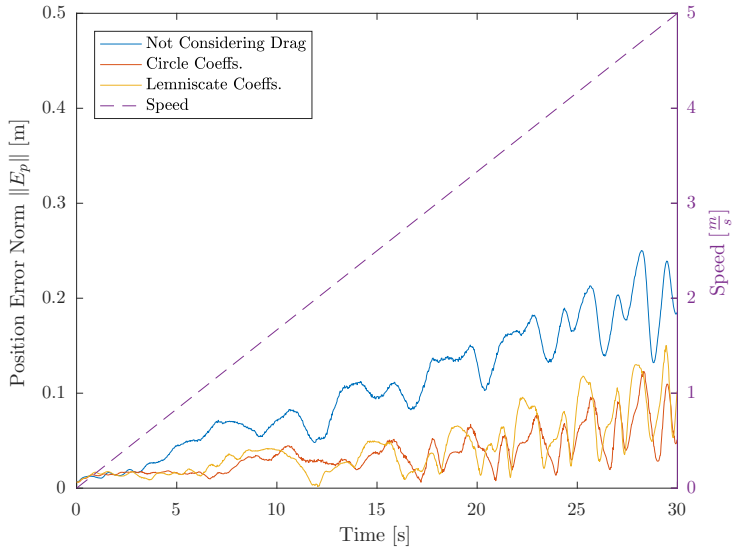


Figure F.6: Position error norm $\|E_p\|$ when ramping the speed on the circle trajectory from $0\,\mathrm{m\,s^{-1}}$ up to $5\,\mathrm{m\,s^{-1}}$ in 30 s without considering rotor drag (solid blue), with drag coefficients estimated on the circle trajectory (solid red), and with drag coefficients estimated on the lemniscate trajectory (solid yellow). The reference speed on the trajectory is shown in dashed purple.

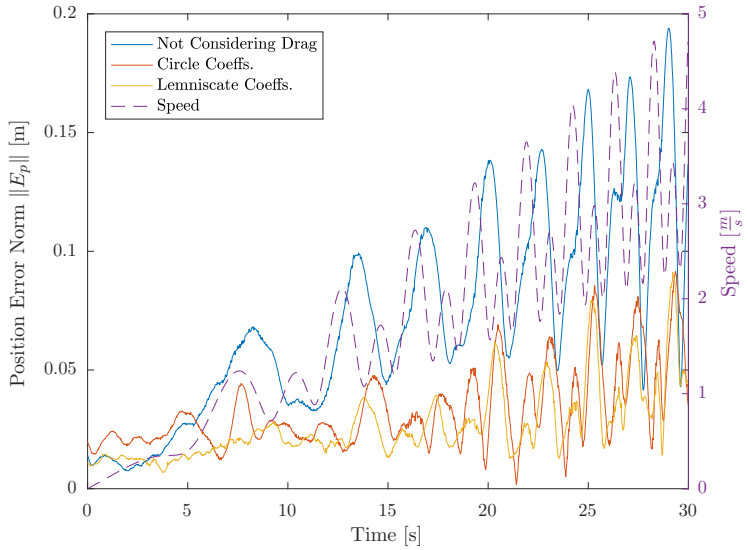


Figure F.7: Position error norm $\|E_p\|$ when ramping the maximum speed on the lemniscate trajectory from $0\,\mathrm{m\,s^{-1}}$ up to $5\,\mathrm{m\,s^{-1}}$ in 30 s without considering rotor drag (solid blue), with drag coefficients estimated on the circle trajectory (solid red), and with drag coefficients estimated on the lemniscate trajectory (solid yellow). The reference speed on the trajectory is shown in dashed purple.

square root of the thrust, which is proportional to the rotor speed, but then assumes the thrust to be constant for the computation of the rotor drag, whereas [127] models the rotor drag to be proportional to the thrust. Simulation results are presented in [76] while real experiments with speeds of up to $2.5\,\mathrm{m\,s^{-1}}$ were conducted in [127].

The controller in [161] considers rotor drag in the computation of the thrust command and the desired orientation but it does not use feed-forward terms on body rates and angular accelerations, which does not allow perfect trajectory tracking. In our work, we use the same thrust model but neglect its dependency on the rotor speed. In [161], also the rotor drag is modeled to depend on the rotor speed, which is physically correct but requires the rotor speeds to be measured for it to be considered in the controller. They present real experiments with speeds of up to $4.0\,\mathrm{m\,s^{-1}}$ and unlike us also show trajectory tracking improvements in vertical flight.

As in our work, [8] considers rotor drag for the computation of the desired thrust, orientation, body rates, and angular accelerations. However, their computations rely on a model where the rotor drag is proportional to the rotor thrust. Also, they neglect the snap of the trajectory and instead require the estimated acceleration and jerk, which are typically not available, for computing the desired body rates and angular accelerations. The presented results in [8] stem from real experiments with speeds of up to $1.0\,\mathrm{m\,s^{-1}}$.

F.9 Conclusion

We proved that the dynamical model of a quadrotor subject to linear rotor drag effects is differentially flat. This property was exploited to compute feed-forward control terms as algebraic functions of a reference trajectory to be tracked. We presented a control policy that uses these feed-forward terms, which compensates for rotor drag effects, and therefore improves the trajectory tracking performance of a quadrotor

Table F.1: Maximum and standard deviation of the position error E_p as well as the absolute trajectory tracking error E_a (F.39) over ten loops on both the circle and the lemniscate trajectory. For each trajectory, we perform the experiment without considering drag, with the drag coefficients identified on the circle trajectory, and with the drag coefficients identified on the lemniscate trajectory.

Trajectory	Params ID	$\max\left(\lVert E_p \rVert\right)$ [cm]	$\sigma\left(\lVert E_p \rVert\right)$ [cm]	E_a [cm]
Circle	Not Cons. Drag	21.08	**2.11**	17.53
	Circle	14.54	2.63	**6.54**
	Lemniscate	**12.39**	2.53	8.16
Lemniscate	Not Cons. Drag	16.79	3.19	11.27
	Circle	10.25	2.30	5.56
	Lemniscate	**10.02**	**2.23**	**5.51**

already from speeds of $0.5\,\mathrm{m\,s^{-1}}$ onwards. The proposed control method reduces the root mean squared tracking error by 50 % independently of the executed trajectory, which we showed by evaluating the tracking performance of a circle and a lemniscate trajectory. In future work, we want to consider the dependency of the rotor drag on the applied thrust to further improve the trajectory tracking performance of quadrotors.

G Low-Level Control, Thrust Mixing, and Saturation

M. Faessler, D. Falanga, and D. Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 476–482. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2640362

Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight

Matthias Faessler, Davide Falanga, and Davide Scaramuzza

Abstract — Quadrotors are well suited for executing fast maneuvers with high accelerations but they are still unable to follow a fast trajectory with centimeter accuracy *without* iteratively *learning* it beforehand. In this paper, we present a novel body-rate controller and an *iterative* thrust-mixing scheme, which improve the trajectory-tracking performance without requiring learning and reduce the yaw control error of a quadrotor, respectively. Furthermore, to the best of our knowledge, we present the first algorithm to cope with motor saturations smartly by *prioritizing* control inputs which are relevant for stabilization and trajectory tracking. The presented body-rate controller uses LQR-control methods to consider both the body rate and the single motor dynamics, which reduces the overall trajectory-tracking error while still rejecting external disturbances well. Our iterative thrust-mixing scheme computes the four rotor thrusts given the inputs from a position-control pipeline. Through the iterative computation, we are able to consider a varying ratio of thrust and drag torque of a single propeller over its input range, which allows applying the desired yaw torque more precisely and hence reduces the yaw-control error. Our prioritizing motor-saturation scheme improves stability and robustness of a quadrotor's flight and may prevent unstable behavior in case of motor saturations. We demonstrate the improved trajectory tracking, yaw-control, and robustness in case of motor saturations in real-world experiments with a quadrotor.

Supplementary Material

A video showing the conducted experiments with a quadrotor is available at:
https://youtu.be/6YEMxFgToyg

G.1 Introduction

G.1.1 Motivation

In recent years, autonomous quadrotors became very popular due to their agility, allowing them to execute aggressive maneuvers. We consider a trajectory to be aggressive if at least one of the quadrotor's motors gets close to saturation during its execution, which is the case if large linear or angular accelerations are required. Even today, when executing such an aggressive trajectory with a quadrotor, the tracking errors without replanning or iteratively learning the maneuver beforehand can be large. For state of the art quadrotor control methods, trajectory tracking errors of up to several body lengths during execution of a fast trajectory (without learning) are reported in e.g. [64] and [108]. Such errors are too large for e.g. fast obstacle avoidance in cluttered environments where iterative learning cannot be applied due to non repetitive motions.

At the heart of precise trajectory tracking with a quadrotor is a body-rate controller that tracks the desired body rates which are typically computed from a high-level control pipeline (e.g. [37], [96]). Only when tracking the desired body rates well, the quadrotor can apply its desired attitude and with that the desired thrust direction precisely, which in turn enables precise translations. To achieve good body-rate tracking, it is crucial to apply the single rotor thrusts precisely, such that the desired body torques and collective thrust can be applied correctly.

To fully exploit the agility of quadrotors, it is desirable to design aggressive trajectories. However, during trajectory design, it is difficult to ensure feasibility while trying to exploit the entire range of feasible motor inputs. And even a trajectory that is feasible with respect to motor saturations cannot guarantee that the tracking controller does not compute motor inputs exceeding its limits due to deviations from the reference trajectory. If such a saturation of one or several motors occurs, the quadrotor may deviate substantially from its reference trajectory or even get unstable if it is not handled correctly.

G.1.2 Contribution

The contribution of this work is threefold. First, we design a novel body-rate controller using LQR methods, which takes the dynamics of the single motors with propellers into account. Compared to previously applied controllers, it improves trajectory tracking

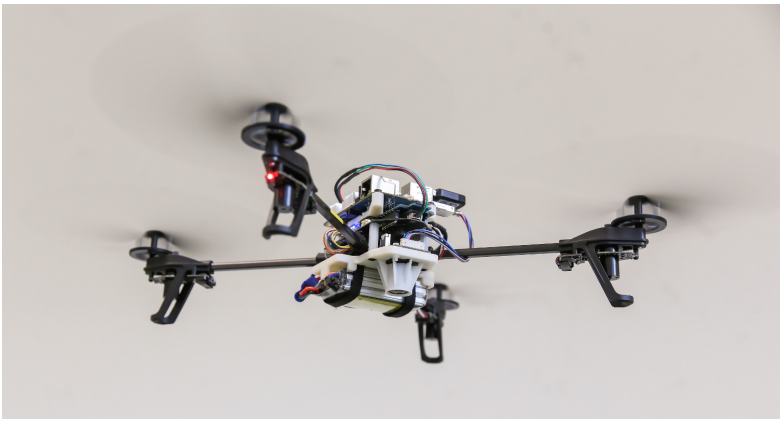


Figure G.1: Our quadrotor used for the experiments of this work.

while maintaining the same disturbance-rejection performance. Second, we improve the computation of the single rotor thrusts such that the desired yaw torque on the quadrotor body, given by a feedback controller, is reached more precisely than with state-of-the-art methods. To do so, we consider that the ratio of thrust and drag torque of a single propeller is not constant over the entire input range, as it is instead assumed in the literature ([37], [96], [110]). Third, we increase the quadrotor's stability and its robustness in case of motor-input saturations. We tackle this by applying a saturation scheme for the single rotor thrusts, which prioritizes between the desired collective thrust and body torques according to their relevance for stabilizing the quadrotor and following a trajectory.

G.1.3 Related Work

Many state-of-the-art quadrotor-control pipelines are making use of a two-level architecture with a high-level position controller and a proportional low-level body-rate controller ([37], [96], [36]), which we also consider in this paper. Besides such cascaded loops of proportional controllers, LQR [16] and nonlinear model predictive control techniques on SO(3) [77] were successfully applied to control the full attitude of quadrotors. In [178], cascaded PID controllers are designed and enhanced with Smith predictors to incorporate the dynamics of the motors for full quadrotor attitude control on SO(3). An LQR attitude controller for a single axis, which is extended with first order dynamics of the motors is presented in [78] and [171]. In contrast, we design an LQR controller for the coupled 3D body rates, incorporating the motor dynamics, which also provides feedback linearization and feed forward on desired angular accelerations.

In both [37] and [96], the low-level control part outputs a desired collective thrust and body torques that need to be applied to the quadrotor's body by the four single rotor thrusts. In other state-of-the-art quadrotor controllers [89], [106], the high-level part

directly outputs the desired collective thrust and body torques. In all these works, the desired collective thrust and body torques need to be converted into four single rotor thrusts which can then be applied by knowing the mapping from motor commands to rotor thrusts, denoted as *thrust mapping*. This is commonly done by solving a system of four equations for the four rotor thrusts, assuming that the ratio of thrust and drag torque of a rotor is constant over its entire motor-input spectrum. We refer to the process of computing the four single rotor thrusts as *thrust mixing*. In this paper, we propose an iterative thrust-mixing scheme that does not require the assumption that the ratio of thrust and drag torque of a rotor is constant, which then allows applying the desired yaw torque more accurately.

Commonly, polynomial trajectories are designed for quadrotors since they are easy to handle mathematically and are dynamically feasible if they are continuous up to a sufficient order of derivatives. They can be generated with a global optimization in which they can be constrained at the start, end, and intermediate waypoints ([106], [139]). Between the waypoints, their feasibility with respect to input limitations cannot be guaranteed. Methods for checking the feasibility of the entire trajectory after generation are proposed in [118] and [66] by simplifying the problem with conservative approximations which do not allow using the full range of feasible inputs. Also, during trajectory execution, it is not guaranteed that the controller does not compute any infeasible inputs since the quadrotor may deviate from a feasible trajectory. Model predictive control methods [117] are able to incorporate input feasibility constraints but are computationally not suitable a low-level controllers running on a micro controller while it is also difficult for them to make use of the entire available range of inputs. Instead of considering feasibility constraints during trajectory generation, in [113] a partial control allocation method that prioritizes the application of desired body torques over collective thrust is used to handle infeasible inputs before applying the motor commands. In this work, we present a saturation scheme that prioritizes control inputs according to their importance for trajectory tracking in case of motor saturations. The presented saturation scheme is able to make use of the full input range of the individual motors.

G.2 System Overview

We consider a quadrotor that is modeled as a rigid body which is controlled by four single rotor thrusts f_i as illustrated in Fig. G.2. By changing these four single rotor thrusts, a three axis torque $\boldsymbol{\eta}$ and a mass normalized collective thrust c can be applied on the quadrotor's body. The relation of the single rotor thrusts to the collective thrust

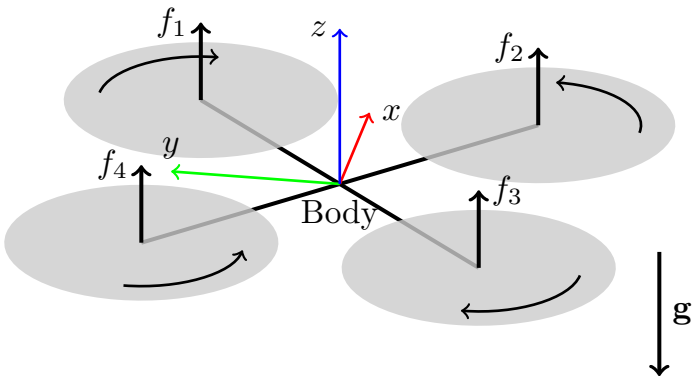


Figure G.2: Quadrotor with coordinate system and single rotor forces.

and the body torques can be formulated using the coordinate system of Fig. G.2 as

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{\sqrt{2}}{2} l(f_1 - f_2 - f_3 + f_4) \\ \frac{\sqrt{2}}{2} l(-f_1 - f_2 + f_3 + f_4) \\ \kappa_1 f_1 - \kappa_2 f_2 + \kappa_3 f_3 - \kappa_4 f_4 \end{bmatrix}, \tag{G.1}$$

$$mc = f_1 + f_2 + f_3 + f_4, \tag{G.2}$$

where l is the quadrotor's arm length, $\kappa_i = \kappa(f_i)$ is a coefficient relating the drag torque and the thrust of a single rotor, and m is the quadrotor's mass. Note that unlike in [96] and [36], we consider the rotor drag torque coefficient κ to be a function of the rotor thrust (cf. Fig G.4) and *not a constant*.

We model the single rotor thrust f and drag torque τ as quadratic polynomials of the motor input u as

$$f(u) = k_2^f u^2 + k_1^f u + k_0^f, \tag{G.3}$$

$$\tau(u) = k_2^\tau u^2 + k_1^\tau u + k_0^\tau, \tag{G.4}$$

where the coefficients k_j^f and k_j^τ are identified by running a single motor with a propeller on a load cell and measuring the resulting forces and moments. The motor input u corresponds to the command we can send to our electronic speed controllers in the range $[-1,\ 1]$. We chose to have three coefficients since it approximates the measured values better than modeling the rotor thrust and drag torque with only a quadratic term, as proposed in e.g. [110]. Figure G.3 compares the two methods for fitting the thrust mapping. From (G.3) and (G.4), we can compute the rotor drag torque coefficient

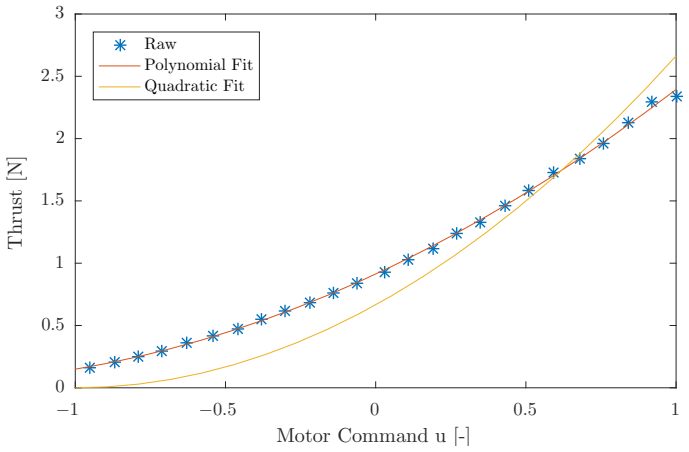


Figure G.3: To approximate the thrust mapping for a single motor with propeller, we fit a second order polynomial into raw thrust measurements obtained by running the motor on a load cell. The polynomial fit approximates the measurements much better than a purely quadratic fit of the form $f(u) = k_2^f u^2$.

as

$$\kappa(f) = \frac{\tau\left(u = \frac{-k_1^f + \sqrt{\left(k_1^f\right)^2 - 4k_2^f\left(k_0^f - f\right)}}{2k_2^f}\right)}{f}. \tag{G.5}$$

Figure G.4 shows the identified values for the rotor drag torque coefficient and how it varies by about 10 % over the entire range of available motor inputs.

As in [96] and [36], we consider that the thrust mapping (G.3) can be refined with rotor-fitness factors γ_i that relate the true thrust f_i and the thrust identified with a load cell $\check{f}_i$ for a certain motor input as $f_i = \gamma_i \check{f}_i$. The rotor fitness factors can be estimated by averaging the applied rotor thrust commands during hover flight.

G.3 Body Rate Control

Due to a hardware architecture with two processing units for high-level and low-level control on our quadrotors, we split the controllers such that the high-level controller computes desired body rates and the low-level controller tracks them. In this section, we present a novel body-rate controller that improves the body-rate-tracking performance by also considering the dynamics of the motors. We achieve this by designing an LQR controller for a dynamical system containing the body rates and body torques as state. The inputs to this controller are the desired body rates $\boldsymbol{\omega}_{des}$ and the desired mass normalized collective thrust c_{des}, which we assume are given from a high-level position

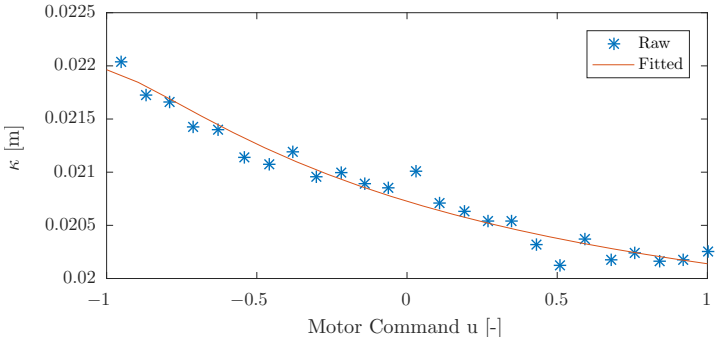


Figure G.4: Values for κ estimated from load cell data obtained by dividing drag torque and thrust values for a motor and propeller as described in Section G.6.1. The fitted curve is the ratio of the fitted quadratic functions for measured thrust and drag torque in (G.3) and (G.4). The range of motor commands is normalized to [-1, 1].

controller (e.g. from [37]).

The dynamics of the quadrotor's body rates $\boldsymbol{\omega}$ are

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1} \cdot (\boldsymbol{\eta} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}), \tag{G.6}$$

where $\boldsymbol{J}$ is the moment of inertia of the quadrotor. Additionally, we model the dynamics of the single rotor thrusts as first order systems

$$\dot{f} = \begin{cases} \frac{1}{\alpha_{up}} (f_{des} - f) & \text{if } f_{des} \geq f \\ \frac{1}{\alpha_{dwn}} (f_{des} - f) & \text{if } f_{des} < f \end{cases}, \tag{G.7}$$

where the time constants α_{up} and α_{dwn} are identified from applying step inputs to a single motor with propeller on a load cell. From these load-cell experiments we found that the single rotor thrust dynamics are considerably different when spinning a motor up or down. Now we can use (G.6) and (G.7) to establish a dynamical system with state $\mathbf{s} = [\boldsymbol{\omega}^T \ \boldsymbol{\eta}^T]^T$ and input $\mathbf{u} = \boldsymbol{\eta}_{des}$. To create a simplified model for the dynamics of the body torques from (G.1) and (G.7), we approximate $\alpha_{up} = \alpha_{dwn} = \alpha$. In practice, we found $\alpha = (\alpha_{up} + \alpha_{dwn})/2$ to be a good approximation which stems from the fact that for changing a body torque, we make use of opposite rotors where one spins up and the other one down. Additionally, we simplify the dynamics of $\boldsymbol{\eta}_z$ by approximating κ to be constant and not depend on the rotor thrust, which leads to the following dynamics of the body torques:

$$\dot{\boldsymbol{\eta}} = \frac{1}{\alpha} (\boldsymbol{\eta}_{des} - \boldsymbol{\eta}). \tag{G.8}$$

Note that the introduced simplifications of these dynamics are necessary for the following feedback-controller design. Linearizing (G.6) and (G.8) around $\boldsymbol{\omega} = \mathbf{0}$ and

$\boldsymbol{\eta} = \mathbf{0}$ leads to the system

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{J}^{-1} \\ \mathbf{0} & -\frac{1}{\alpha}\mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\eta} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \frac{1}{\alpha}\mathbf{I}_3 \end{bmatrix} \boldsymbol{\eta}_{des}, \tag{G.9}$$

which we can use to design an infinite-horizon LQR control law $\mathbf{u} = -\mathbf{K}_{lqr}\mathbf{s}$ that minimizes the cost function

$$\int \mathbf{s}^T\mathbf{Q}\mathbf{s} + \mathbf{u}^T\mathbf{R}\mathbf{u}\, dt, \tag{G.10}$$

where $\mathbf{Q}$ is a diagonal weight matrix and $\mathbf{R}$ is the identity matrix. The solution to the formulated LQR problem is a gain matrix of the form

$$\mathbf{K}_{lqr} = \begin{bmatrix} k_{\omega_{xy}} & 0 & 0 & k_{\eta_{xy}} & 0 & 0 \\ 0 & k_{\omega_{xy}} & 0 & 0 & k_{\eta_{xy}} & 0 \\ 0 & 0 & k_{\omega_z} & 0 & 0 & k_{\eta_z} \end{bmatrix}, \tag{G.11}$$

which corresponds to a PD controller of the body rates. Additionally, we add feed forward terms such that $\boldsymbol{\omega}_{des}$ is reached with $\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}}_{des}$, resulting in the control policy

$$\boldsymbol{\eta}_{des} = \mathbf{K}_{lqr} \begin{bmatrix} \boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}} \\ \boldsymbol{\eta}_{ref} - \hat{\boldsymbol{\eta}} \end{bmatrix} + \hat{\boldsymbol{\omega}} \times \boldsymbol{J}\hat{\boldsymbol{\omega}} + \boldsymbol{J}\dot{\boldsymbol{\omega}}_{des}, \tag{G.12}$$

with $\boldsymbol{\eta}_{ref} = \boldsymbol{\omega}_{des} \times \boldsymbol{J}\boldsymbol{\omega}_{des} + \boldsymbol{J}\dot{\boldsymbol{\omega}}_{des}$ computed from (G.6). The vector $\hat{\boldsymbol{\omega}}$ are the estimated body rates measured by the onboard gyroscopes, and $\hat{\boldsymbol{\eta}}$ are the estimated body torques obtained by estimating the single rotor thrusts with (G.7) and using (G.1) to transform them into body torques. Note that this estimation makes use of the non-symmetric model (G.7) of the rotor thrusts, i.e., $\alpha_{up} \neq \alpha_{dwn}$, while for the control design we assumed $\alpha_{up} = \alpha_{dwn}$. Also note that this estimation can be improved if feedback of the rotor speeds is available. In the controller (G.12), the term $\hat{\boldsymbol{\omega}} \times \boldsymbol{J}\hat{\boldsymbol{\omega}}$ provides feedback linearization, compensating for the coupling terms in the body-rate dynamics, and the term $\boldsymbol{J}\dot{\boldsymbol{\omega}}_{des}$ can be used as feed forward on desired angular accelerations that can be computed from a trajectory to be tracked due to the quadrotor's differential flatness property [106].

G.4 Iterative Thrust Mixing

To compute the single rotor thrusts that achieve the desired body torques $\boldsymbol{\eta}_{des}$ and collective thrust c_{des}, which we denote as *mixer inputs*, we have to solve (G.1) and (G.2) for f_i. Since we consider the rotor drag torque coefficients to be a function of the rotor thrust, we cannot solve this system of equations directly, but we can do so iteratively.

To initialize the iteration, we start by setting the single rotor thrusts equal such that

they achieve the desired collective thrust:

$$f_i = \frac{mc_{des}}{4}. \tag{G.13}$$

Note that these values are only used to compute the rotor drag torque coefficients in the first iteration. Then, we start the iteration with the following two steps: i) solve (G.5) to get κ_i, and ii) solve (G.1) and (G.2) with $\boldsymbol{\eta}_{des}$ and c_{des} for f_i. Additionally to quadrotors, this iterative scheme can also be applied to other multirotors.

G.5 Saturation with Input Priorities

Once we have computed the desired single rotor thrusts, we have to make sure that they lie within the feasible range $[f_{min}, f_{max}]$ for each single motor. Naively, this can be done by clipping each rotor thrust if its desired value is outside this range. This is a simple and fast procedure with the drawback that none of the desired mixer inputs $\boldsymbol{\eta}_{des}$ and c_{des} is achieved exactly if one of the rotor thrusts is clipped. Nonetheless, not all these mixer inputs are equally important in terms of the quadrotor's ability to stabilize and track a trajectory. Since a quadrotor can only produce a collective thrust in its body upwards direction, it has to be aligned with the desired acceleration for following a trajectory in 3D space. The rotation around the thrust direction is irrelevant for the translational motion of the quadrotor. Therefore, we want to give least priority to achieve the desired yaw torque in case of an input saturation. On the other hand, the quadrotor uses roll and pitch torques to change its thrust direction which enables stabilization and therefore makes them the most important inputs. Furthermore, state of the art control methods for quadrotors (e.g. [37], [96]) are based on the assumption that the orientation of the thrust vector can be changed quickly. For these reasons, in case of an input saturation, we want to give highest priority to applying the desired roll and pitch torques, second highest priority to applying the desired collective thrust, and lowest priority to applying the desired yaw torque.

G.5.1 Yaw-Torque Saturation

We achieve this prioritization by a saturation scheme as summarized in Algorithm 4. First, the single rotor thrusts are computed according to Section G.4. If one of the single rotor thrusts exceeds its limits, we try to change the applied yaw torque to avoid saturation, given that the desired yaw torque is above a certain minimum $\eta_{z,assured}$, which we can optionally impose. Such an assured yaw torque might be desired for applications where we want to guarantee that we always have some control on the heading of a quadrotor. In case of saturation, we do not apply the iterative mixer in order to save time since we are unable to apply the desired yaw torque anyways. To do the yaw-torque saturation, we find the rotor that violates the input limit the most,

Algorithm 4 Rotor Thrust Saturation

Compute f_i as detailed in Section G.4
Perform yaw-torque saturation:
if Motor saturated **AND** $|\eta_{z,des}| > \eta_{z,assured}$ **then**
  Find rotor j that violates thrust limits the most
  $f_j \leftarrow \gamma_j \cdot f_{limit}$
  Solve (G.1) and (G.2) for $f_{i \setminus j}$ and η_z
  if $\text{sign}(\eta_{z,des}) \cdot \eta_z < \eta_{z,assured}$ **then**
    $\eta_z \leftarrow \text{sign}(\eta_{z,des}) \cdot \eta_{z,assured}$
    Solve (G.1) and (G.2) for f_i
Perform collective-thrust saturation:
if Motor saturated **then**
  if NOT(upper **AND** lower saturation reached) **then**
    Find rotor j that violates thrust limits the most
    Shift f_i equally s.t. $f_j = \gamma_j \cdot f_{limit}$
Enforce single rotor thrust limits by thrust clipping

set it to the corresponding limit and then solve (G.1) and (G.2) for the remaining rotor thrusts and the yaw torque. If the resulting yaw torque is still above the value we want to assure, we successfully enforced all the rotor-thrust limits by only changing the applied yaw torque. In other words, in this case, Algorithm 4 guarantees that the quadrotor applies the desired roll and pitch torques and the desired collective thrust, but *not* the desired yaw torque. If the resulting yaw torque is below the value we want to assure, we set it to the assured value $\eta_{z,assured}$ and recompute the rotor thrusts.

G.5.2 Collective-Thrust Saturation

If one of the rotor-thrust limits is still violated, we try to change the applied collective thrust to avoid saturation. This is only possible if two rotors do not violate the upper and the lower limit simultaneously, in which case it is impossible to achieve the desired roll and pitch torques by changing the applied collective thrust. If only one limit is violated, we find the rotor that violates the input limit the most, set it to its limit and shift the remaining rotor thrusts by the same amount. In this case, Algorithm 4 ensures that the quadrotor applies the desired roll and pitch torques but *not* the desired collective thrust and *not* the desired yaw torque.

G.5.3 Thrust Clipping

At this point, if a rotor still violates its input limits, we have to apply thrust clipping and can therefore not achieve any of the desired mixer inputs precisely. Note that the presented procedure uses the full range of available thrusts for each rotor also

considering individual rotor fitness factors.

G.6 Experiments

G.6.1 Experimental Setup

We built our quadrotor from selected off-the-shelf components and custom 3D printed parts (see Fig. G.1) with a total weight of 503 g. It is based on the frame of the Parrot AR.Drone 2.0 including their motors, motor controllers, gears, and propellers. On this frame, we use a PX4IOAR adapter board and a PX4FMU autopilot that runs all the presented algorithms of this paper. All the details about our drone can be found in [36].

We identified the thrust mapping (G.3) and the torque mapping (G.4) by putting one single motor with propeller on a *ATI Mini40* load cell. The first order time constants $\alpha_{up} = 11\,\text{ms}$ and $\alpha_{dwn} = 27\,\text{ms}$ were estimated by applying step inputs to a motor on the load cell.

The following three subsections provide results of experiments that show the effects of applying our LQR body-rate controller, iterative mixer, and prioritizing saturation, respectively, as isolated as possible. All the flight experiments were conducted in an OptiTrack motion capture system to acquire a ground truth state measurement of the quadrotor. A video of the conducted experiments can be found on https://youtu.be/6YEMxFgToyg.

G.6.2 Body-Rate Controller

We compare the proposed LQR body-rate controller to a proportional controller from our previous work [37] (identical to [96]) in its disturbance rejection and trajectory-tracking performance. Our goal for the LQR controller is to maintain the disturbance-rejection performance of the previously used proportional controller but achieve better trajectory tracking. Note that when neglecting the motor dynamics in the LQR design, we obtain a proportional controller and hence compare a similar controller which either considers motor dynamics or not. We conduct all experiments for a low gain proportional controller (a), as used in [37], a high gain proportional controller (b), and the proposed LQR controller (c) with a proportional gain corresponding to the one of (b).

We measure the disturbance-rejection performance by hitting one of the quadrotor's arms upwards during hover flight and measuring the time it takes it to settle again.[1]This is illustrated in Fig. G.5, where the low gain proportional and the LQR controller have almost identical settling times and the high gain proportional controller has a significantly higher one. We measure the trajectory-tracking performance by flying a

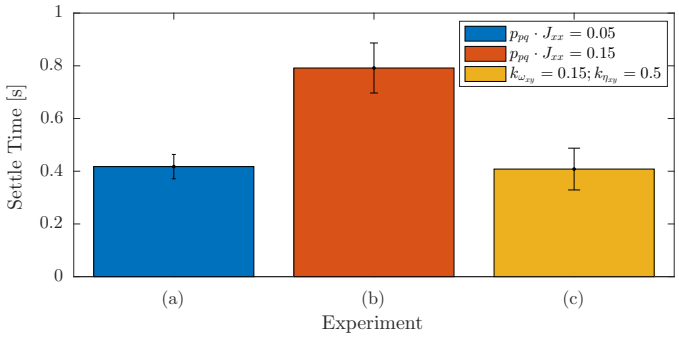


Figure G.5: Disturbance-rejection performance measured by the time it takes the quadrotor to settle after disturbing it by hitting one of the arms upwards (averaged over 10 disturbances). The experiments (a) and (b) are conducted with different gains using the body-rate controller from [37] or [96] and (c) is conducted with the proposed LQR controller. The vertical black lines indicate the standard deviation of the settling time.

quadrotor back and forth in the x-direction with a maximum acceleration of $12\,\mathrm{m\,s^{-2}}$ and a maximum velocity of $5.7\,\mathrm{m\,s^{-1}}$ on a trajectory composed of multiple polynomial segments[1] that are continuous in snap (cf. Fig. G.6) and measuring the tracking errors illustrated in Fig. G.7. It shows that increasing the gain of the proportional controller improves trajectory tracking significantly. Our proposed LQR controller shows a slightly larger position error than the high gain proportional controller but still reduces it noticeably by more than 25 % compared to the low gain proportional controller. To provide a fair comparison of the two controllers, we apply the proposed iterative mixer and prioritizing saturation (only shortly active) in all the experiments.

Table G.1 summarizes the results of the disturbance rejection and trajectory-tracking experiments to compare the three different controllers. It especially shows that increasing the gain of the proportional controller improves trajectory tracking but also reduces its disturbance-rejection performance significantly [from (a) to (c)]. On the other hand, the LQR controller (c) with equivalent gains to (b) can almost improve trajectory tracking as much but maintains the same disturbance-rejection performance of the low gain proportional controller (a).

G.6.3 Iterative Mixer

Figure G.8 compares the performance of the proposed iterative mixer and a one-shot mixer in a three minute hover flight. The corresponding statistics in Table G.2 show that the iterative mixer reduces the maximum, mean, and standard deviation of the yaw error by more than 35 %. In this comparison, both methods make use of the same polynomial model for the thrust and torque mapping (G.3), (G.4), respectively. We conducted this experiment in hover flight to suppress non modeled dynamics and motor saturations in order to isolate the effect of the iterative mixer. Note that in

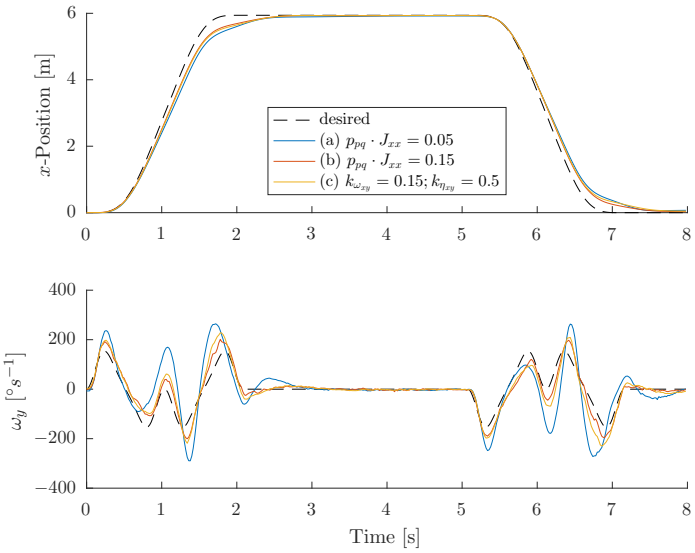


Figure G.6: Position and body-rate-tracking performance for a back-and-forth motion in the x-direction. The trajectory is composed of multiple polynomial segments with maximum acceleration of $12\,\mathrm{m\,s^{-2}}$ and maximum velocity of $5.7\,\mathrm{m\,s^{-1}}$. The experiments (a) and (b) are conducted with different gains using the body-rate controller from [37] or [96] and (c) is conducted with the proposed LQR controller. The corresponding errors are illustrated in Fig. G.7.

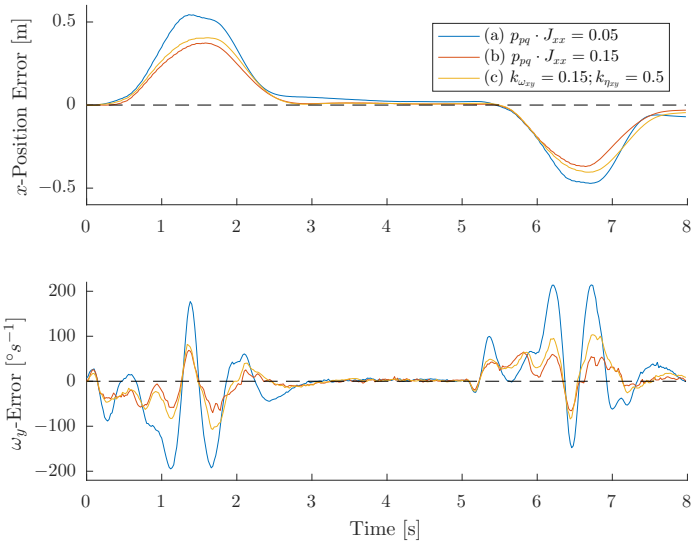


Figure G.7: Tracking errors for the trajectory in Fig. G.6. The corresponding error statistics are presented in Table G.1, where we also compare the trajectory tracking and the disturbance-rejection performance of each controller.

Table G.1: Position error in x-direction (e_x), thrust-direction error angle (θ), and pitch-rate error (e_{ω_y}) statistics for the experiments illustrated in Fig. G.6 and averaged over twelve back-and-forth motions, as well as settle time (t_{settle}) statistics for the experiment illustrated in Fig. G.5.

	(a)	(b)	(c)
$\mu(e_x)$ [cm]	**14.50**	**9.39**	**10.35**
$\sigma(e_x)$ [cm]	23.05	15.60	17.19
$\mu(\theta)$ [°]	5.55	3.84	4.14
$\sigma(\theta)$ [°]	7.33	5.00	5.22
$\mu(e_{\omega_y})$ [° s^{-1}]	42.28	14.88	21.52
$\sigma(e_{\omega_y})$ [° s^{-1}]	72.38	23.65	32.41
$\mu(t_{settle})$ [s]	**0.42**	**0.79**	**0.41**
$\sigma(t_{settle})$ [s]	0.05	0.09	0.08

Table G.2: Yaw error statistics comparison between a standard one-shot mixer and the proposed iterative mixer in hover flight.

	One-Shot	**Iterative**	
Max Yaw Error	8.371	**5.209**	[°]
Mean Yaw Error	2.667	**1.462**	[°]
Yaw Error Standard Deviation	2.549	**1.593**	[°]

non-hover flight, where larger roll and pitch torques are required, the iterative mixer becomes more advantageous due to larger commanded rotor-thrust differences. In practice, after the third mixer iteration, the error between the desired and achieved yaw torque becomes negligible ($\ll 1\,\%$) for different tested motor types. The error made by a one-shot mixer can be more than 5 % of the desired yaw torque. Note that this error depends on the differences of the commanded single rotor thrusts and vanishes when the four commanded thrusts are equal.

G.6.4 Prioritizing Saturation

The performance of the prioritizing saturation compared to thrust clipping is shown in Fig. G.9 with an experiment[1] where a quadrotor is commanded to do a simultaneous step in height by 1 m and yaw by 90°. In this experiment, the minimum and maximum single rotor thrust were artificially set to 0.8 N and 1.6 N respectively (hover thrust: 1.25 N) to force the saturation scheme to become active without requiring an aggressive

[1] See video on: https://youtu.be/6YEMxFgToyg

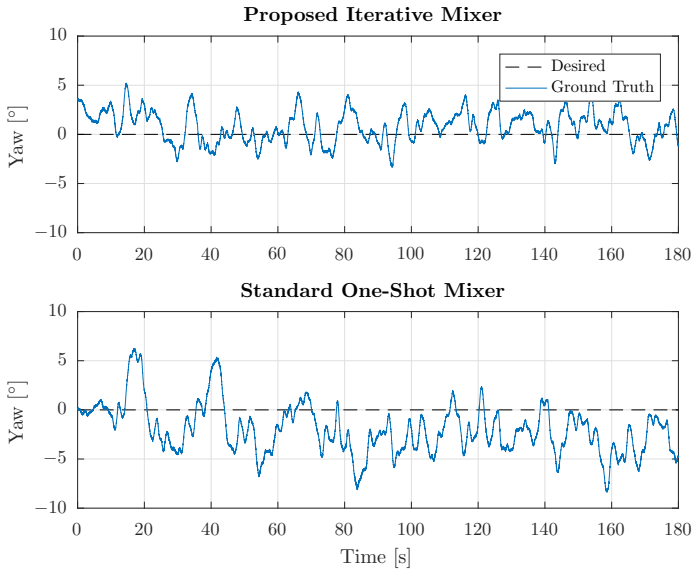


Figure G.8: Comparison of the actual and desired yaw angle during hover flight using the proposed iterative mixer (top) and a one-shot mixer (bottom). Statistics of the yaw error are depicted in Table G.2.

maneuver. It can be seen that with the prioritizing saturation, the x and y position stay close to their constant desired values, whereas they deflect a lot from the desired value in case of thrust clipping. This even causes the quadrotor to crash into the ground for one run when thrust clipping is applied. On the other hand, with the prioritizing saturation, compared to thrust clipping, it takes longer for the yaw angle to settle on the desired value since its priority is the smallest. Note that when a saturation occurs, we do not make use of the iterative mixer since we are unable to reach the desired yaw torque due to the saturation. The computation times of the different steps in the presented saturation scheme are summarized in Table G.3.

Table G.3: Iterative mixing and saturation computation times. As comparison, the thrust mixing with a one-shot mixer takes 1.4 µs with 0.3 µs standard deviation.

	Mean [µs]	Standard Deviation [µs]
Thrust Mixing (3 iter.)	243.7	5.8
Yaw Saturation	166.6	4.1
Thrust Saturation	1.3	0.3
Thrust Clipping	1.5	0.4
Total	413.1	7.1

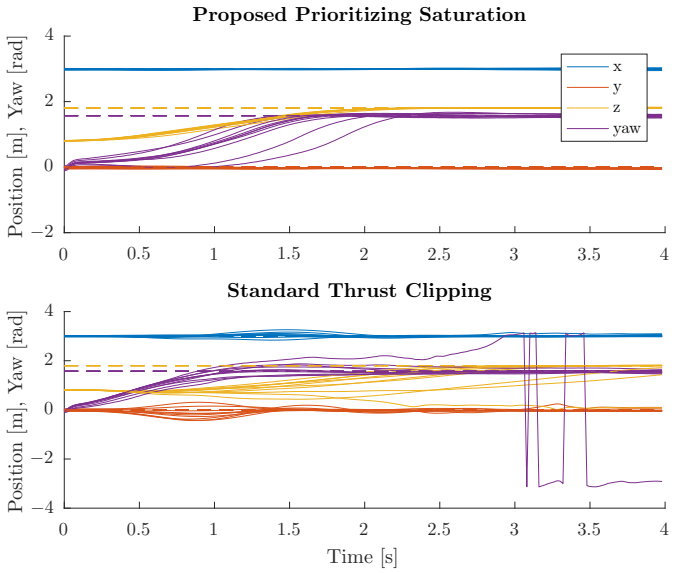


Figure G.9: Response in position and yaw for ten runs where the quadrotor simultaneously performs a 1 m step in height and a 90° step in yaw starting at $t = 0\,\text{s}$ using prioritizing saturation (top) and thrust clipping (bottom). The reference values after the step are: $x = 3.0\,\text{m}$, $y = 0.0\,\text{m}$, $z = 1.8\,\text{m}$, and $yaw = ß/2\,\text{rad}$, and the assured yaw torque is $\eta_{z,assured} = 0.01\,\text{N m}$. Solid lines show ground truth data and dashed lines show desired reference values.

G.7 Conclusion

We presented an LQR based body-rate controller, an iterative mixer to compute the desired single rotor thrusts, and a prioritizing motor-saturation scheme which we all evaluated in real experiments with a quadrotor in a motion capture system. Compared to the state-of-the-art proportional controllers, the proposed body-rate controller almost halved the body-rate tracking error and reduced the position error by more than 25 % during a fast trajectory while preserving its damping properties against external disturbances. The iterative mixer reduces the yaw error by considering a non constant ratio of thrust and drag torque of a single rotor by more than 35 % in hover flight. The presented motor saturation scheme prioritizes roll and pitch, which are most important for stabilization and trajectory tracking, before collective thrust and yaw. We demonstrated that through this prioritization, the stability and robustness of a quadrotor is increased in case of motor-input saturations.

Bibliography

[1] P. Abbeel, A. Coates, and A. Y. Ng. "Autonomous Helicopter Aerobatics through Apprenticeship Learning". In: *Int. J. Robot. Research* 29.13 (June 2010), pp. 1608–1639. DOI: 10.1177/0278364910371999.

[2] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy. "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments". In: *SPIE Conf. on Unmanned Systems Technology*. 2009.

[3] S. Ahrens, D. Levine, G. Andrews, and J. How. "Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2009.

[4] D. Alejo, J. A. Cobano, G. Heredia, and A. Ollero. "Collision-Free 4D Trajectory Planning in Unmanned Aerial Vehicles for Assembly and Structure Construction". In: *J. Intell. Robot. Syst.* 73.1-4 (Oct. 2013), pp. 783–795. DOI: 10.1007/s10846-013-9948-x.

[5] G. Allibert, D. Abeywardena, M. Bangura, and R. Mahony. "Estimating body-fixed frame velocity and attitude from inertial measurements for a quadrotor vehicle". In: *IEEE Conf. Control Appl. (CCA)*. Oct. 2014, pp. 978–983. DOI: 10.1109/cca.2014.6981462.

[6] A. Bachrach, R. He, and N. Roy. "Autonomous Flight in Unstructured and Unknown Indoor Environments". In: *Eur. Micro Aerial Vehicle Conf.* Delft, the Netherlands, Sept. 2009.

[7] A. Bachrach, S. Prentice, R. He, P. Henry, A. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy. "Estimation, Planning and Mapping for Autonomous Flight Using an RGB-D Camera in GPS-denied Environments". In: *J. Field Robot.* 31.11 (Sept. 2012), pp. 1320–1343.

[8] M. Bangura. "Aerodynamics and Control of Quadrotors". PhD thesis. College of Engineering and Computer Science, The Australian National University, 2017.

[9] M. Bangura and R. Mahony. "Real-time Model Predictive Control for Quadrotors". In: *IFAC World Congress* 47.3 (2014), pp. 11773–11780. DOI: 10.3182/20140824-6-za-1003.00203.

[10] M. Bangura and R. Mahony. "Thrust Control for Multirotor Aerial Vehicles". In: *IEEE Trans. Robot.* 33.2 (Apr. 2017), pp. 390–405. DOI: 10.1109/tro.2016.2633562.

[11] M. Bangura, R. Mahony, H. Lim, and H. J. Kim. "An open-source implementation of a unit quaternion based attitude and trajectory tracking for quadrotors". In: *Australasian Conf. Robot. Autom.* 2014.

[12] M. Bangura, M. Melega, R. Naldi, and R. Mahony. "Aerodynamics of rotor blades for quadrotors". In: *arXiv e-prints* (Dec. 2016). URL: http://arxiv.org/abs/1601.00733.

[13] B. M. Bell and F. W. Cathey. "The Iterated Kalman Filter Update as a Gauss-Newton Method". In: *IEEE Trans. Autom. Control* 38.2 (1993), pp. 294–297. DOI: 10.1109/9.250476.

[14] R. Bischoff, U. Huggenberger, and E. Prassler. "KUKA youBot - a mobile manipulator for research and education". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011.

[15] M. Bloesch, S. Weiss, D. Scaramuzza, and R. Siegwart. "Vision Based MAV Navigation in Unknown and Unstructured Environments". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2010.

[16] S. Bouabdallah, A. Noth, and R. Siegwart. "PID vs LQ control techniques applied to an indoor micro quadrotor". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Vol. 3. Sept. 2004, pp. 2451–2456. DOI: 10.1109/IROS.2004.1389776.

[17] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[18] A. Breitenmoser, L. Kneip, and R. Siegwart. "A monocular vision-based system for 6D relative robot localization". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Sept. 2011. DOI: 10.1109/iros.2011.6094851.

[19] D. Brescianini, M. Hehn, and R. D'Andrea. *Nonlinear Quadrocopter Attitude Control*. Tech. rep. Department of Mechanical and Process Engineering, ETHZ, Oct. 2013.

[20] D. Brescianini, M. Hehn, and R. D'Andrea. "Quadrocopter Pole Acrobatics". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Nov. 2013, pp. 3472–3479. DOI: 10.1109/IROS.2013.6696851.

[21] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher. "Fast Global Minimization of the Active Contour/Snake Model". In: *Journal of Mathematical Imaging and Vision* 28.2 (2007).

[22] P. J. Bristeau, P. Martin, E. Salaün, and N. Petit. "The role of propeller aerodynamics in the model of a quadrotor UAV". In: *IEEE Eur. Control Conf. (ECC)*. Aug. 2009, pp. 683–688.

[23] R. Brockers, M. Hummenberger, S. Weiss, and L. Matthies. "Towards Autonomous Navigation of Miniature UAV". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. June 2014. DOI: 10.1109/cvprw.2014.98.

[24] M. Burri, M. Bloesch, Z. Taylor, R. Siegwart, and J. Nieto. "A framework for maximum likelihood parameter identification applied on MAVs". In: *J. Field Robot.* (June 2017), pp. 1–18. DOI: 10.1002/rob.21729.

[25] M. Burri, M. Dätwiler, M. W. Achtelik, and R. Siegwart. "Robust state estimation for Micro Aerial Vehicles based on system dynamics". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 5278–5283. DOI: 10.1109/ICRA.2015.7139935.

[26] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza. "Low-latency localization by Active LED Markers tracking using a Dynamic Vision Sensor". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013. DOI: 10.1109/IROS.2013.6696456.

[27] A. Chambolle and T. Pock. "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging". In: *Journal of Mathematical Imaging and Vision* 40.1 (2011).

[28] N. Chaturvedi, A. Sanyal, and N. H. McClamroch. "Rigid-Body Attitude Control". In: *IEEE Control Systems* 31.3 (June 2011), pp. 30–51. DOI: 10.1109/mcs.2011.940459.

[29] P. Chen and Y. Hwang. "Practical path planning among movable obstacles". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 1991.

[30] C. D. Crousaz, F. Farshidian, M. Neunert, and J. Buchli. "Unified Motion Control for Dynamic Quadrotor Maneuvers Demonstrated on Slung Load and Rotor Failure Tasks". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 2223–2229. DOI: 10.1109/ICRA.2015.7139493.

[31] M. Cutler and J. P. How. "Analysis and Control of a Variable-Pitch Quadrotor for Agile Flight". In: *J. Dynamic Syst. Meas. Control* 137.10 (July 2015), p. 101002. DOI: 10.1115/1.4030676.

[32] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar. "Devices, systems, and methods for automated monitoring enabling precision agriculture". In: *IEEE Int. Conf. Automation Science and Engineering (CASE)*. Aug. 2015, pp. 462–469. DOI: 10.1109/coase.2015.7294123.

[33] E. Eade and T. Drummond. "Unified loop closing and recovery for real time monocular SLAM". In: *British Machine Vis. Conf. (BMVC)*. 2008.

[34] J. Engel, J. Sturm, and D. Cremers. "Camera-Based Navigation of a Low-Cost Quadrocopter". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2012.

[35] M. Faessler, D. Falanga, and D. Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 476–482. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2640362.

[36] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor MAV". In: *J. Field Robot.* 33.4 (2016), pp. 431–450. ISSN: 1556-4967. DOI: 10.1002/rob.21581.

[37] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza. "Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 1722–1729. DOI: 10.1109/ICRA.2015.7139420.

[38] M. Faessler, A. Franchi, and D. Scaramuzza. "Detailed Derivations of 'Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories'". In: *arXiv e-prints* (Dec. 2017). URL: http://arxiv.org/abs/1712.02402.

[39] M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 620–626. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2776353.

[40] M. Faessler, E. Mueggler, K. Schwabe, and D. Scaramuzza. "A Monocular Pose Estimation System based on Infrared LEDs". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 907–913. DOI: 10.1109/ICRA.2014.6906962.

[41] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017. DOI: 10.1109/icra.2017.7989679.

[42] J. Ferrin, R. Leishman, R. Beard, and T. McLain. "Differential flatness based control of a rotorcraft for aggressive maneuvers". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Sept. 2011, pp. 2688–2693. DOI: 10.1109/iros.2011.6095098.

[43] M. Fiala. "ARTag, a Fiducial Marker System Using Digital Techniques". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. June 2005. DOI: 10.1109/cvpr.2005.74.

[44] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24.6 (1981), pp. 381–395. ISSN: 0001-0782. DOI: http://doi.acm.org/10.1145/358669.358692.

[45] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. "Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013, pp. 3962–3970. DOI: 10.1109/IROS.2013.6696923.

[46] C. Forster, M. Pizzoli, and D. Scaramuzza. "Air-Ground Localization and Map Augmentation Using Monocular Dense Reconstruction". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013, pp. 3971–3978. DOI: 10.1109/IROS.2013.6696924.

[47] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza. "Continuous On-Board Monocular-Vision–based Aerial Elevation Mapping for Quadrotor Landing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 111–118. URL: http://dx.doi.org/10.1109/ICRA.2015.7138988.

[48] C. Forster, M. Pizzoli, and D. Scaramuzza. "Appearance-based Active, Monocular, Dense Depth Estimation for Micro Aerial Vehicles". In: *Robotics: Science and Systems (RSS)*. 2014. DOI: 10.15607/RSS.2014.X.029.

[49] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584.

[50] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems". In: *IEEE Trans. Robot.* 33.2 (2017), pp. 249–265. DOI: 10.1109/TRO.2016.2623335.

[51] A. Franchi and A. Mallet. "Adaptive closed-loop speed control of BLDC motors with applications to multi-rotor aerial vehicles". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017, pp. 5203–5208. DOI: 10.1109/icra.2017.7989610.

[52] E. Fresk and G. Nikolakopoulos. "Full quaternion based attitude control for a quadrotor". In: *IEEE Eur. Control Conf. (ECC)*. July 2013, pp. 3864–3869.

[53] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Muller-Bessler, and B. Huhnke. "Up to the limits: Autonomous Audi TTS". In: *IEEE Intell. Vehicles Symp.* June 2012, pp. 541–547. DOI: 10.1109/ivs.2012.6232212.

[54] J. Gallier. *Geometric Methods and Applications: For Computer Science and Engineering*. Springer Publishing Company, Incorporated, 2011. DOI: 10.1007/978-1-4419-9961-0.

[55] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. "Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. June 2007. DOI: 10.1109/cvpr.2007.383245.

[56] M. Garzon, J. Valente, D. Zapata, and A. Barrientos. "An Aerial-Ground Robotic System for Navigation and Obstacle Mapping in Large Outdoor Areas". In: *Sensors* 13.1 (2013), pp. 1247–1267.

[57] R. Gill and R. D'Andrea. "Propeller thrust and drag in forward flight". In: *IEEE Conf. Cont. Tech. Applications (CCTA)*. Aug. 2017, pp. 73–79. DOI: 10.1109/ccta.2017.8062443.

[58] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella. "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots". In: *IEEE Robot. Autom. Lett.* 1.2 (July 2016), pp. 661–667. ISSN: 2377-3766. DOI: 10.1109/LRA.2015.2509024.

[59] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey. "Supporting wilderness search and rescue using a camera-equipped mini UAV". In: *J. Field Robot.* 25.1-2 (2007), pp. 89–110. DOI: 10.1002/rob.20226.

[60] V. Grabe, M. Riedel, H. H. Bulthoff, P. R. Giordano, and A. Franchi. "The TeleKyb framework for a modular and extendible ROS-based quadrotor control". In: *Eur. Conf. Mobile Robots (ECMR)*. Sept. 2013, pp. 19–25. DOI: 10.1109/ecmr.2013.6698814.

[61] M. Hamer, M. Waibel, and R. D'Andrea. "Knowledge transfer for high-performance quadrocopter maneuvers". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Nov. 2013, pp. 1714–1719. DOI: 10.1109/iros.2013.6696580.

[62] A. Handa, R. A. Newcombe, A. Angeli, and A. J. Davison. "Real-Time Camera Tracking: When is High Frame-Rate Best?" In: *Eur. Conf. Comput. Vis. (ECCV)*. 2012. DOI: 10.1007/978-3-642-33786-4_17.

[63] M. Hehn and R. D'Andrea. "A flying inverted pendulum". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2011, pp. 763–770. DOI: 10.1109/icra.2011.5980244.

[64] M. Hehn and R. D'Andrea. "A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers". In: *J. Mechatronics* 24.8 (2014), pp. 954–965. ISSN: 0957-4158. DOI: http://dx.doi.org/10.1016/j.mechatronics.2014.09.013.

[65] M. Hehn and R. D'Andrea. "An iterative learning scheme for high performance, periodic quadrocopter trajectories". In: *IEEE Eur. Control Conf. (ECC)*. July 2013, pp. 1799–1804.

[66] M. Hehn and R. D'Andrea. "Real-Time Trajectory Generation for Quadrocopters". In: *IEEE Trans. Robot.* 31.4 (2015), pp. 877–892.

[67] M. Hehn, R. Ritz, and R. D'Andrea. "Performance benchmarking of quadrotor systems using time-optimal control". In: *Auton. Robots* 33.1-2 (Mar. 2012), pp. 69–88. DOI: 10.1007/s10514-012-9282-3.

[68] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin. "Quadrotor helicopter flight dynamics and control: Theory and experiment". In: *AIAA Guidance, Navigation, and Control Conference*. Vol. 2. Aug. 2007, p. 4.

[69] S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts. "Combined optic-flow and stereo-based navigation of urban canyons for a UAV". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2005.

[70] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung, D. F. Wolf, G. S. Sukhatme, and D. C. MacKenzie. "Adaptive teams of autonomous aerial and ground robots for situational awareness". In: *J. Field Robot.* 24.11–12 (2007), pp. 991–1014. DOI: 10.1002/rob.20222.

[71] H. Huang, G. Hoffmann, S. Waslander, and C. Tomlin. "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2009, pp. 3277–3282.

[72] M. Irani and P. Anandan. "All About Direct Methods". In: *Proc. Workshop Vis. Algorithms: Theory Pract.* 1999, pp. 267–277.

[73] A. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero, and R. Cano. "Control of an aerial robot with multi-link arm for assembly tasks". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2013, pp. 4916–4921. DOI: 10.1109/icra.2013.6631279.

[74] S. G. Johnson. *The NLopt nonlinear-optimization package*. URL: http://ab-initio.mit.edu/nlopt.

[75] M. Kaess, A. Ranganathan, and F. Dellaert. "iSAM: Incremental Smoothing and Mapping". In: *IEEE Trans. Robot.* 24.6 (Dec. 2008), pp. 1365–1378.

[76] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel. "Nonlinear feedback control of Quadrotors exploiting First-Order Drag Effects". In: *IFAC World Congress*. Vol. 50. 1. July 2017, pp. 8189–8195. DOI: 10.1016/j.ifacol.2017.08.1267.

[77] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. "Fast nonlinear model predictive control for multicopter attitude tracking on SO(3)". In: *IEEE Conf. Control Appl. (CCA)*. Sept. 2015, pp. 1160–1166. DOI: 10.1109/CCA.2015.7320769.

[78] Y. Kawai and K. Uchiyama. "Design of frequency shaped LQR considering dynamic characteristics of the actuator". In: *IEEE Int. Conf. Unmanned Aircraft Syst. (ICUAS)*. June 2016, pp. 1235–1239. DOI: 10.1109/ICUAS.2016.7502630.

[79] B. Keiser. "Torque Control of a KUKA youBot Arm". MA thesis. University of Zurich, 2013.

[80] H. K. Khalil. *Nonlinear Systems*. 2nd ed. Prentice-Hall, 1996.

[81] G. Klein and D. Murray. "Parallel tracking and mapping for small AR workspaces". In: *IEEE ACM Int. Sym. Mixed and Augmented Reality (ISMAR)*. Nara, Japan, Nov. 2007, pp. 225–234.

[82] L. Kneip, D. Scaramuzza, and R. Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2011, pp. 2969–2976. DOI: 10.1109/CVPR.2011.5995464.

[83] K. Kondak, K. Krieger, A. Albu-Schaeffer, M. Schwarzbach, M. Laiacker, I. Maza, A. Rodriguez-Castano, and A. Ollero. "Closed-Loop Behavior of an Autonomous Helicopter Equipped with a Robotic Arm for Aerial Manipulation Tasks". In: *Int. J. Adv. Robot. Syst.* 10.2 (Jan. 2013), p. 145. DOI: 10.5772/53754.

[84] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. URL: https://books.google.ch/books?id=4rKaGwAACAAJ.

[85] G.-J. Kruijff, V. Tretyakov, T. Linder, F. Pirri, M. Gianni, P. Papadakis, M. Pizzoli, A. Sinha, E. Pianese, S. Corrao, F. Priori, S. Febrini, and S. Angeletti. "Rescue robots at earthquake-hit Mirandola, Italy: A field report". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. 2012.

[86] S. Kumar and R. Gill. "Path following for quadrotors". In: *IEEE Conf. Cont. Tech. Applications (CCTA)*. Aug. 2017, pp. 2075–2081. DOI: 10.1109/ccta.2017.8062759.

[87] M. Langerwisch, T. Wittmann, S. Thamke, T. Remmersmann, A. Tiderko, and B. Wagner. "Heterogeneous Teams of Unmanned Ground and Aerial Robots for Reconnaissance and Surveillance - A Field Experiment". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. 2013.

[88] V. A. Laurense, J. Y. Goh, and J. C. Gerdes. "Path-tracking for autonomous vehicles at the limit of friction". In: *IEEE Am. Control Conf. (ACC)*. May 2017, pp. 5586–5591. DOI: 10.23919/acc.2017.7963824.

[89] T. Lee, M. Leoky, and N. McClamroch. "Geometric tracking control of a quadrotor UAV on SE(3)". In: *IEEE Conf. Decision Control (CDC)*. Dec. 2010, pp. 5420–5425. DOI: 10.1109/CDC.2010.5717652.

[90] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain. "Quadrotors and Accelerometers: State Estimation with an Improved Dynamic Model". In: *Control Systems, IEEE* 34.1 (Feb. 2014), pp. 28–41. ISSN: 1066-033X. DOI: 10.1109/MCS.2013.2287362.

[91] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig. "Deep neural networks for improved, impromptu trajectory tracking of quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017, pp. 5183–5189. DOI: 10.1109/icra.2017.7989607.

[92] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor". In: *IEEE J. Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[93] V. Lippiello, G. Loianno, and B. Siciliano. "MAV indoor navigation based on a closed-form solution for absolute scale velocity estimation using Optical Flow and inertial data". In: *IEEE Conf. Decision Control (CDC)*. 2011.

[94] G. Loianno, C. Brunner, G. McGrath, and V. Kumar. "Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 404–411. DOI: 10.1109/lra.2016.2633290.

[95] S. Lupashin and R. D'Andrea. "Adaptive fast open-loop maneuvers for quadrocopters". In: *Auton. Robots* 33.1-2 (Apr. 2012), pp. 89–102. DOI: 10.1007/s10514-012-9289-9.

[96] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea. "A platform for aerial robotics research and demonstration: The Flying Machine Arena". In: *J. Mechatronics* 24.1 (Feb. 2014), pp. 41–54. DOI: 10.1016/j.mechatronics.2013.11.006.

[97] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea. "A simple learning strategy for high-speed quadrocopter multi-flips". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2010, pp. 1642–1648. DOI: 10.1109/ROBOT.2010.5509452.

[98] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart. "A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013.

[99] E. Lyu, Y. Lin, W. Liu, and M. Q.-H. Meng. "Vision based autonomous gap-flying-through using the micro unmanned aerial vehicle". In: *IEEE Can. Conf. on Electrical and Computer Engineering (CCECE)*. May 2015. DOI: 10.1109/ccece.2015.7129368.

[100] R. Mahony, V. Kumar, and P. Corke. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor". In: *IEEE Robot. Autom. Mag.* 19.3 (2012), pp. 20–32. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2206474.

[101] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, and N. Tomatis. "The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. 2012.

[102] P. Martin and E. Salaün. "The true role of accelerometer feedback in quadrotor control". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2010, pp. 1623–1629. DOI: 10.1109/robot.2010.5509980.

[103] A. Martinelli. "Vision and IMU Data Fusion: Closed-Form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination". In: *IEEE Trans. Robot.* 28.1 (Feb. 2012), pp. 44–60. DOI: 10.1109/tro.2011.2160468.

[104] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel. "Quaternion-Based Hybrid Control for Robust Global Attitude Tracking". In: *IEEE Trans. Autom. Control* 56.11 (Nov. 2011), pp. 2555–2566. DOI: 10.1109/tac.2011.2108490.

[105] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision". In: *Auton. Robots* 33.1–2 (2012), pp. 21–39.

[106] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[107] D. Mellinger, N. Michael, and V. Kumar. "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors". In: *Int. Symp. Experimental Robotics (ISER)*. Dec. 2010.

[108] D. Mellinger, N. Michael, and V. Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors". In: *Int. J. Robot. Research* 31.5 (2012), pp. 664–674. DOI: 10.1177/0278364911434236.

[109] L. Merino, F. Caballero, J. R. Martínez-de-Dios, I. Maza, and A. Ollero. "An Unmanned Aircraft System for Automatic Forest Fire Monitoring and Measurement". In: *J. Intell. Robot. Syst.* 65.1-4 (Aug. 2011), pp. 533–548. DOI: 10.1007/s10846-011-9560-x.

[110] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. "The GRASP multiple micro UAV testbed". In: *IEEE Robot. Autom. Mag.* 17.3 (Sept. 2010), pp. 56–65.

[111] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. "Collaborative mapping of an earthquake-damaged building via ground and aerial robots". In: *J. Field Robot.* 29.5 (Sept. 2012), pp. 832–841.

[112] G. Michieletto, M. Ryll, and A. Franchi. "Control of statically hoverable multirotor aerial vehicles and application to rotor-failure robustness for hexarotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017, pp. 2747–2752. DOI: 10.1109/icra.2017.7989320.

[113] J. C. Monteiro, F. Lizarralde, and L. Hsu. "Optimal control allocation of quadrotor UAVs subject to actuator constraints". In: *IEEE Am. Control Conf. (ACC)*. July 2016, pp. 500–505. DOI: 10.1109/ACC.2016.7524963.

[114] E. Mueggler, M. Faessler, F. Fontana, and D. Scaramuzza. "Aerial-guided Navigation of a Ground Robot among Movable Obstacles". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. 2014, pp. 1–8. DOI: 10.1109/SSRR.2014.7017662.

[115] M. Mueller, S. Lupashin, and R. D'Andrea. "Quadrocopter ball juggling". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2011, pp. 4972–4978. DOI: 10.1109/IROS.2012.6385963.

[116] M. W. Mueller and R. D'Andrea. "Relaxed hover solutions for multicopters: Application to algorithmic redundancy and novel vehicles". In: *Int. J. Robot. Research* 35.8 (Oct. 2015), pp. 873–889. DOI: 10.1177/0278364915596233.

[117] M. W. Mueller and R. D'Andrea. "A model predictive controller for quadrocopter state interception". In: *IEEE Eur. Control Conf. (ECC)*. July 2013, pp. 1383–1389.

[118] M. W. Mueller, M. Hehn, and R. D'Andrea. "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation". In: *IEEE Trans. Robot.* 31.6 (2015), pp. 1294–1310.

[119] Y. Mulgaonkar, G. Cross, and V. Kumar. "Design of small, safe and robust quadrotor swarms". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2015, pp. 2208–2215. DOI: 10.1109/icra.2015.7139491.

[120] R. R. Murphy. *Disaster Robotics*. The MIT Press, Cambridge, MA, 2014.

[121] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma. "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots". In: *J. Field Robot.* 30.1 (2013), pp. 44–63.

[122] A. Nash, K. Daniel, S. Koenig, and A. Felner. "Theta*: Any-Angle Path Planning on Grids". In: *AAAI Conf. Artificial Intell.* 2007.

[123] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. DOI: 10.1093/comjnl/7.4.308.

[124] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2016. DOI: 10.1109/icra.2016.7487274.

[125] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. "DTAM: Dense Tracking and Mapping in Real-Time". In: *Int. Conf. Comput. Vis. (ICCV)*. Nov. 2011, pp. 2320–2327.

[126] E. Olson. "AprilTag: A Robust and Flexible Visual Fiducial System". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2011.

[127] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel. "Nonlinear control of VTOL UAVs incorporating flapping dynamics". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Nov. 2013, pp. 2419–2425. DOI: 10.1109/iros.2013.6696696.

[128] T. Ozaslan, G. Loianno, J. Keller, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood. "Autonomous Navigation and Mapping for Inspection of Penstocks and Tunnels With MAVs". In: *IEEE Robot. Autom. Lett.* 2.3 (July 2017), pp. 1740–1747. DOI: 10.1109/lra.2017.2699790.

[129] F. Pasqualetti, A. Franchi, and F. Bullo. "On Cooperative Patrolling: Optimal Trajectories, Complexity Analysis, and Approximation Algorithms". In: *IEEE Trans. Robot.* 28.3 (June 2012), pp. 592–606. DOI: 10.1109/tro.2011.2179580.

[130] D. Perez, I. Maza, F. Caballero, D. Scarlatti, E. Casado, and A. Ollero. "A Ground Control Station for a Multi-UAV Surveillance System". In: *J. Intell. Robot. Syst.* 69.1-4 (Aug. 2012), pp. 119–130. DOI: 10.1007/s10846-012-9759-5.

[131] T. Perkins and R. R. Murphy. "Active and mediated opportunistic cooperation between an unmanned aerial vehicle and an unmanned ground vehicle". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. Oct. 2013. DOI: 10.1109/ssrr.2013.6719376.

[132] T. Pintaric and H. Kaufmann. "A rigid-body target design methodology for optical pose-tracking systems". In: *Proc. ACM symposium on Virtual reality software and technology VRST*. 2008. DOI: 10.1145/1450579.1450594.

[133] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 2609–2616. URL: http://dx.doi.org/10.1109/ICRA.2014.6907233.

[134] P. Pounds, R. Mahony, and P. Corke. "Modelling and Control of a Quad-Rotor Robot". In: *Australasian Conf. Robot. Autom.* 2006.

[135] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar. "Influence of aerodynamics and proximity effects in quadrotor flight". In: *Int. Symp. Experimental Robotics (ISER)*. 2013, pp. 289–302.

[136] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System". In: *ICRA Workshop Open Source Softw.* Vol. 3. 2. 2009, p. 5.

[137] G. V. Raffo, M. G. Ortega, and F. R. Rubio. "MPC with Nonlinear H_∞ Control for Path Tracking of a Quad-Rotor Helicopter". In: *IFAC World Congress* 41.2 (2008), pp. 8564–8569. DOI: 10.3182/20080706-5-kr-1001.01448.

[138] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard. "LQR control for a quadrotor using unit quaternions: Modeling and simulation". In: *Int. Conf. on Electronics, Communications and Computing (CONIELECOMP)*. Mar. 2013, pp. 172–178. DOI: 10.1109/conielecomp.2013.6525781.

[139] C. Richter, A. Bry, and N. Roy. "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments". In: *Proc. Int. Symp. Robot. Research (ISRR)*. 2013, pp. 1–8.

[140] R. Ritz, M. Hehn, S. Lupashin, and R. D'Andrea. "Quadrocopter performance benchmarking using optimal control". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Sept. 2011, pp. 5179–5186. DOI: 10.1109/iros.2011.6094775.

[141] E. Rosten, R. Porter, and T. Drummond. "Faster and Better: A Machine Learning Approach to Corner Detection". In: *IEEE Trans. Pattern Anal. Machine Intell.* 32.1 (Jan. 2010), pp. 105–119. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.275.

[142] L. I. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1–4 (1992).

[143] F. Ruffier and N. Franceschini. "Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2004.

[144] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi. "6D physical interaction with a fully actuated aerial robot". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017, pp. 5190–5195. DOI: 10.1109/icra.2017.7989608.

[145] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. B. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss. "Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments". In: *IEEE Robot. Autom. Mag.* (2014).

[146] D. Scaramuzza and F. Fraundorfer. "Visual Odometry [Tutorial]. Part I: The First 30 Years and Fundamentals". In: *IEEE Robot. Autom. Mag.* 18.4 (Dec. 2011), pp. 80–92. ISSN: 1070-9932. DOI: 10.1109/MRA.2011.943233.

[147] K. Schmid, P. Lutz, T. Tomic, E. Mair, and H. Hirschmuller. "Autonomous Vision-based Micro Air Vehicle for Indoor and Outdoor Navigation". In: *J. Field Robot.* 31 (4 2014), pp. 537–570.

[148] A. P. Schoellig, F. L. Mueller, and R. D'Andrea. "Optimization-based iterative learning for precise quadrocopter trajectory tracking". In: *Auton. Robots* 33.1-2 (Apr. 2012), pp. 103–127. DOI: 10.1007/s10514-012-9283-2.

[149] A. P. Schoellig, C. Wiltsche, and R. D'Andrea. "Feed-forward parameter identification for precise periodic quadrocopter motions". In: *IEEE Am. Control Conf. (ACC)*. June 2012, pp. 4313–4318. DOI: 10.1109/acc.2012.6315248.

[150] M. Schulz, F. Augugliaro, R. Ritz, and R. D'Andrea. "High-speed, steady flight with a quadrocopter in a confined environment using a tether". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Sept. 2015, pp. 1279–1284. DOI: 10.1109/iros.2015.7353533.

[151] S. Shen. "Autonomous Navigation in Complex Indoor and Outdoor Environments with Micro Aerial Vehicles". PhD thesis. Philadelphia, PA, USA: University of Pennsylvania, July 2014.

[152] S. Shen, N. Michael, and V. Kumar. "Autonomous indoor 3D exploration with a micro-aerial vehicle". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012.

[153] S. Shen, N. Michael, and V. Kumar. "Autonomous multi-floor indoor navigation with a computationally constrained MAV". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2011, pp. 20–25. DOI: 10.1109/ICRA.2011.5980357.

[154] S. Shen, N. Michael, and V. Kumar. "Stochastic differential equation-based exploration algorithm for autonomous indoor 3D exploration with a micro aerial vehicle". In: *J. Field Robot.* 31.12 (Oct. 2012), pp. 1431–1444.

[155] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. "Vision-based state estimation and trajectory control towards aggressive flight with a quadrotor". In: *Robotics: Science and Systems (RSS)*. June 2013. DOI: 10.15607/RSS.2013.IX.03.

[156] M. Stilman and J. J. Kuffner. "Navigation among movable obstacles: Real-time reasoning in complex environments". In: *Int. J. Humanoid Robot.* 2.4 (2005), pp. 479–504.

[157] M. Stilman, K. Nishiwaki, S. Kagami, and J. J. Kuffner. "Planning and Executing Navigation Among Movable Obstacles". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)* (2006).

[158] H. Strasdat, J. Montiel, and A. Davison. "Real-time Monocular SLAM: Why Filter?" In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2010.

[159] J. Stühmer, S. Gumhold, and D. Cremers. "Real-Time Dense Geometry from a Handheld Camera". In: *Pattern Recognition*. Vol. 6376. Lecture Notes in Computer Science. 2010, pp. 11–20. ISBN: 978-3-642-15985-5. DOI: 10.1007/978-3-642-15986-2_2.

[160] S. Suzuki and K. Abe. "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734-189X. DOI: http://dx.doi.org/10.1016/0734-189X(85)90016-7.

[161] J. Svacha, K. Mohta, and V. Kumar. "Improving quadrotor trajectory tracking by compensating for aerodynamic effects". In: *IEEE Int. Conf. Unmanned Aircraft Syst. (ICUAS)*. June 2017, pp. 860–866. DOI: 10.1109/icuas.2017.7991501.

[162] R. Szeliski and D. Scharstein. "Sampling the disparity space image". In: *IEEE Trans. Pattern Anal. Machine Intell.* 26.3 (2004), pp. 419–425.

[163] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer, 2010. ISBN: 9781848829343.

[164] S. Tang and V. Kumar. "Mixed integer Quadratic Program Trajectory Generation for a Quadrotor With a Cable-Suspended Payload". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 2216–2222. DOI: 10.1109/ICRA.2015.7139492.

[165] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. "Stanley: The robot that won the DARPA Grand Challenge". In: *J. Field Robot.* 23.9 (2006), pp. 661–692. DOI: 10.1002/rob.20147.

[166] M. Tognon, A. Testa, E. Rossi, and A. Franchi. "Takeoff and landing on slopes via inclined hovering with a tethered aerial robot". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2016, pp. 1702–1707. DOI: 10.1109/iros.2016.7759273.

[167] N. Trawny and S. I. Roumeliotis. *Indirect Kalman Filter for 3D Attitude Estimation*. Tech. rep. 2005-002. University of Minnesota, Dept. of Comp. Sci. & Eng., Mar. 2005.

[168] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. "Bundle Adjustment – A Modern Synthesis". In: *Vision Algorithms: Theory and Practice*. Ed. by W. Triggs, A. Zisserman, and R. Szeliski. Vol. 1883. LNCS. Springer Verlag, 2000, pp. 298–372.

[169] S. Umeyama. "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns". In: *IEEE Trans. Pattern Anal. Machine Intell.* 13.4 (1991).

[170] G. Vogiatzis and C. Hernández. "Video-based, Real-Time Multi View Stereo". In: *Image Vis. Comput.* 29.7 (2011), pp. 434–441. DOI: 10.1016/j.imavis.2011.01.006.

[171] B. Wang, K. A. Ghamry, and Y. Zhang. "Trajectory tracking and attitude control of an unmanned quadrotor helicopter considering actuator dynamics". In: *IEEE Chin. Control Conf. (CCC)*. July 2016, pp. 10795–10800. DOI: 10.1109/ChiCC.2016.7555068.

[172] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart. "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012.

[173] S. Weiss, D. Scaramuzza, and R. Siegwart. "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments". In: *J. Field Robot.* 28.6 (Nov. 2011), pp. 854–874.

[174] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart. "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium". In: *J. Field Robot.* 30.5 (Aug. 2013), pp. 803–831. DOI: 10.1002/rob.21466.

[175] A. Wendel, M. Maurer, G. Graber, T. Pock, and H. Bischof. "Dense reconstruction on-the-fly". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2012.

[176] M. Werlberger, T. Pock, and H. Bischof. "Motion Estimation with Non-Local Total Variation Regularization". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2010.

[177] G. Wilfong. "Motion panning in the presence of movable obstacles". In: *Proc. ACM Symp. Computat. Geometry*. 1988.

[178] Y. Yu, S. Yang, M. Wang, C. Li, and Z. Li. "High performance full attitude control of a quadrotor on SO(3)". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2015, pp. 1698–1703. DOI: 10.1109/ICRA.2015.7139416.

[179] B. Yüksel, G. Buondonno, and A. Franchi. "Differential Flatness and Control of Protocentric Aerial Manipulators with Any Number of Arms and Mixed Rigid-/Elastic-Joints". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2016, pp. 561–566. DOI: 10.1109/IROS.2016.7759109.

[180] S. Zhou, M. K. Helwa, and A. P. Schoellig. "Design of Deep Neural Networks as Add-on Blocks for Improving Impromptu Trajectory Tracking". In: *CoRR* abs/1705.10932 (2017). eprint: 1705.10932. URL: http://arxiv.org/abs/1705.10932.

[181] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart. "MAV Navigation through Indoor Corridors Using Optical Flow". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2010.

[182] J.-C. Zufferey and D. Floreano. "Fly-inspired visual steering of an ultralight indoor aircraft". In: *IEEE Trans. Robot.* 22.1 (Feb. 2006), pp. 137–146.

Matthias Faessler

Curriculum Vitae

Andreasstrasse 15, AND 2.16
8050 Zurich, Switzerland
☎ *+41 44 635 43 41*
✉ *faessler@ifi.uzh.ch*

Education

Dec 2012–Apr 2018 **Ph.D. Program in Computer Science**, *Robotics and Perception Group, University of Zurich (UZH)*, Zurich, Switzerland.
- Research topics are in robotics, control, and perception aware path planning.
- Supervisor: Prof. Dr. Davide Scaramuzza

Jun 2012 **Master of Science in Mechanical Engineering**, *Swiss Federal Institute of Technology (ETH)*, Zurich, Switzerland.
- Focus on robotics and control.

Mar 2011–Jan 2012 **Visiting Research Scholar at the General Robotics, Automation, Sensing and Perception (GRASP) Lab**, *University of Pennsylvania (Penn)*, Philadelphia, PA, USA.
- Development of control algorithms for small coaxial helicopters.

Aug 2010 **Bachelor of Science in Mechanical Engineering**, *Swiss Federal Institute of Technology (ETH)*, Zurich, Switzerland.
- Focus on mechatronics.

Theses

Mar–Oct 2011 **Master Thesis: Modeling, Control and Trajectory Tracking with a CoaX helicopter**, *GRASP Lab, University of Pennsylvania (Penn)*, Philadelphia, PA, USA.
- Design of a dynamical model and a novel, nonlinear control for a coaxial helicopter.
- Development of an algorithm to build trajectories in an environment with obstacles.
- Advisor: Prof. Dr. Vijay Kumar

Feb–Jul 2010 **Semester Thesis: Aggressive Juggling Height Transitions with the Blind Juggler**, *Institute for Dynamic Systems and Control (IDSC), ETH*, Zurich, Switzerland.
- Maximized juggling performance based on precise model allowing high-performance open-loop juggling height changes.
- Advisor: Prof. Dr. Raffaello D'Andrea

Sep 2009–Jan 2010 **Bachelor Thesis: IMU-Based Hovering with a Micro Helicopter**, *Autonomous Systems Lab (ASL), ETH*, Zurich, Switzerland.
- Development and implementation of a control strategy for rejecting wind gusts with a micro coaxial helicopter.
- Advisor: Prof. Dr. Roland Siegwart

Professional Experience

Feb–Jun 2012 **Industrial Internship in Systems Engineering**, *Rheinmetall Air Defence*, Zurich, Switzerland.
- Creation of a temperature simulation for the turret of an Oerlikon Gun and its data processing system in particular.
- Design of a concept for a turret ventilation of an Oerlikon Gun.
- Error estimations for different methods of measuring the twist between two systems.

Feb–Mar 2007 **Practical Workshop Internship**, *Pamasol Willi Mäder AG*, Pfäffikon SZ, Switzerland.
- Practical experience with drilling, milling, lathing, welding and metal processing in general.

2002–2007 **Several Temporary Employments**.
- Private lessons in mathematics, car mechanic, carpenter, plumber.

Other Activities

2009–Present **Voluntary work in a soccer club**, *FC Feusisberg-Schindellegi*, Feusisberg, Switzerland.
- Occasional substitute coach and camp leader.
- Member of the organizational committee of our annual amateur soccer tournament.
- Development of a software for automatically planning and evaluating tournaments.

Summer Schools and Subsidiary Courses

May-Jun 2016 **Arbeits- und Leistungsprobleme überwinden und Kreativität fördern**, *University of Zurich*.

Feb-May 2016 **Startup Campus CTI Entrepreneurship Training, Module 2: Business Concept**, *University of Zurich*.

Feb-Mar 2016 **Managing conflicts**, *University of Zurich*.

Sep-Oct 2014 **How to give a presentation effectively and persuasively?**, *University of Zurich*.

Nov 2013 **Wissenschaft öffentlich kommunizieren**, *University of Zurich*.
- Writing for mass media.
- Speaking in front of a camera.

Jul 2013 **International Computer Vision Summer School (ICVSS)**, Calabria, Italy.

Teaching Experience

Student Supervision

2016 **Semester Thesis of Kevin Egger**.
- On-board Height Estimation for Quadrotors

2015 **Master Thesis of Raphael Meyer**.
- Design of a Custom Quadrotor Platform

2015 **Master Thesis of Michael Gassner**.
- Aggressive Maneuvers with Quadrotors

2014 **Master Thesis of Astrid Schlestein**.
- System Identification and Model Based State Estimation for Quadrotors

2014 **Semester Thesis of Maximilian Schulz**.
- Robust Emergency Procedures for Quadrotors

2014 **Semester Thesis of Adrian Rechy Romero**.
- Self-Calibration for Quadrotors

2014 **Semester Thesis of Raphael Meyer**.
- Design of Custom Interface Electronics for Quadrotors

2013 **Master Thesis of Benjamin Keiser**.
- Torque Control of a KUKA youBot Arm.
- Winner of KUKA Best Student Project Award 2013.
- Video: https://youtu.be/OMZ1XVXErKY

2013 **Master Thesis of Karl Schwabe**.
- A Monocular Pose Estimation System based on Infrared LEDs.
- Video: https://youtu.be/8Ui3MoOxcPQ

Teaching Assistance

Sep–Dec 2010 **Teaching Assistant for Mechanics I**, *Institute of Mechanical Systems (IMES), ETH*, Zurich, Switzerland.
- Preparation and supervision of exercises and exams (plus grading).
- Supervisor: Prof. Dr. Edoardo Mazza

Exhibitions, Demos, and Videos

Sep 2017 **IROS 2017 Autonomous Drone Racing Challenge**, Vancouver, Canada.
- Video: https://youtu.be/ImQ8izGQzuk

Sep 2015 **Scientifica**, Zurich, Switzerland.
- Presenting research of the Robotics and Perception Group.
- 25,000 Visitors

May 2015 **SwissCore 20th Anniversary**, Brussels, Belgium.
- Research demonstration to EU and SNSF officials, including Jean-Pierre Bourguignon (ERC president), Martin Vetterli (President of the Research Council of the SNSF) and Michael Hengartner (President of the University of Zurich)

Jun 2014 **KUKA Innovation Award**, Zurich, Switzerland.
- Presenting our robot demo at Automatica in Munich.
- Video: https://youtu.be/OFPv3BegbFg

Aug 2013 **Scientifica**, Zurich, Switzerland.
- Presenting research of the Robotics and Perception Group.
- 20,000 Visitors

Apr 2013 **Festival de Robotique**, Lausanne, Switzerland.
- Presenting research of the Robotics and Perception Group.
- Video: https://youtu.be/Uiwrl-Fot30
- 17,000 Visitors

Mar 2013 **Easter Video**.
- Video: https://youtu.be/1GXQjrmLeX4

Mar 2013 **Robots on Tour**, Zurich, Switzerland.
- Presenting research of the Robotics and Perception Group.
- Guiding visitors through exhibitions of various international robotics research groups.
- 4000 Visitors

Awards

Jun 2014 **KUKA Innovation Award 2014**.
- 20 000€ prize money.

2014 **SSRR Best Paper Award Finalist**.

Attended Conferences

Sep 2017 **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, Vancouver, Canada.

Jun 2017 **IEEE International Conference on Robotics and Automation (ICRA)**, Singapore, Singapore.

May 2015 **IEEE International Conference on Robotics and Automation (ICRA)**, Seattle, WA, USA.

Sep 2015 **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, Hamburg, Germany.

May 2013 **IEEE International Conference on Robotics and Automation (ICRA)**, Karlsruhe, Germany.

Publications

Journals

[1] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza, **Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories**, *IEEE Robotics and Automation Letters*, 2018.

[2] Matthias Faessler, Davide Falanga, and Davide Scaramuzza, **Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight**, *IEEE Robotics and Automation Letters*, 2017.

[3] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza, **Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor Micro Aerial Vehicle**, *Journal of Field Robotics*, 2016.

[4] Alessandro Giusti, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella, **A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots**, *IEEE Robotics and Automation Letters*, 2016.

Peer-Reviewed Proceedings

[1] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza, **Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing**, *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2017, Singapore.

[2] Matthias Faessler, Flavio Fontana, Christian Forster, and Davide Scaramuzza, **Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor**, *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2015, Seattle.

[3] Christian Forster, Matthias Faessler, Flavio Fontana, Manuel Werlberger, and Davide Scaramuzza, **Continuous On-Board Monocular-Vision-based Elevation Mapping Applied to Autonomous Landing of Micro Aerial Vehicles**, *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2015, Seattle.

[4] Elias Mueggler, Matthias Faessler, Flavio Fontana, and Davide Scaramuzza, **Aerial-guided Navigation of a Ground Robot among Movable Obstacles**, *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2014, Toyako-cho.

[5] Matthias Faessler, Elias Mueggler, Karl Schwabe, and Davide Scaramuzza, **A Monocular Pose Estimation System based on Infrared LEDs**, *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014, Hong Kong.

Technical Reports

[1] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza, **Detailed Derivations of "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories"**, *ArXiv e-prints*, arXiv:1712.02402, 2017.

Other Publications

[1] Matthias Faessler, Elias Mueggler, and Davide Scaramuzza, **Happy Easter from the AI Lab**, *Video Competition at International Joint Conferences on Artificial Intelligence (IJCAI)*, 2013, Beijing.